Development of an Integrated Fuel Management System with the aid of CPLDs

By

S SWANEPOEL

Study Leader: Prof PJ Bakkes

This thesis is submitted as the requirement for a Masters Degree in Engineering at the

University of Stellenbosch

Electric and Electronic Engineering Department

November 2000

Declaration

I, the undersigned, hereby declare that the work contained in this thesis is my own original work and that I have not previously in its entirety or in part submitted it at any university for a degree.

Signature: Date:

Abstract

The need for a locally manufactured, cost-effective, fuel management system led to the design and development of a functional prototype.

This thesis presents the design, development and full implementation of two functional prototypes. While field tests performed on the first prototype assisted in identifying necessary modifications, a need for greater complexity in support hardware was also identified. In order to comply with the cost-effective nature of the project, it was realized that this complexity could only be achieved via the implementation of a CPLD based system. Using Altera's Maxplus II design software, the CPLD code was synthesized according to the required specifications then simulated and analyzed.

On completion of the new CPLD based system, the second prototype, one of Altera's megacore functions is implemented and used as a substitute to an external hardware. All necessary modifications were successfully completed and the system was installed.

Uittreksel

Die benodiging van 'n lokale vervaardigde, koste effektiewe brandstofbeheerstelsel het gelei tot die ontwerp en ontwikkeling van 'n funksionele prototipe.

Hierdie tesis dek die ontwerp, ontwikkeling en volle implementasie van twee funksionele prototipes. Terwyl toetse op die eerste prototipe gebruik is om die nodige aanpassings te identifiseer, is daar ook besef dat daar meer komplekse hardeware onderstuening benodig word.

Om die koste effektiewe aard van die projek te handhaaf, is daar gesien dat die nodige kompleksieteit alleenlik deur middel van die implementering van 'n 'CPLD' gebaseerde sisteem bereik kan word.
Deur gebruik te maak van Altera se Maxplus II ontwerp sagteware, is die 'CPLD' kode met die nodige spesifiekasies gesintiseer, gesimuleer en geanaliseer.

Na voltooing van die tweede prototiepe, die 'CPLD' gebaseerde sisteem, is een van Altera se megacore funksies geimplementeer en gebruik as 'n plaasvervanger vir eksterne hardware. Alle nodige aanpassings is suksesvol voltooi en die sisteem is geinstalleer.

Acknowledgments

Thanks be to God for providing me with the opportunity and strength required for the completion of this thesis. I would also like to thank Prof PJ Bakkes for his guidance throughout the duration of this thesis.

A special acknowledgment to Altera for establishing the University program which allowed me the opportunity of using their most current technology in this thesis.

Table of Contents

CHAPTER 3 Design of a Phase 1 Micrcontroller Based System using Gate-level Logic Components

CHAPTER 4 Design of a PHASE 2 Microcontroller Based System using Complex Programmable Logic Device Technology

List of Figures

List of Tables

Important Definitions

AHDL	:	Altera Hardware Description Language. A design entry language which supports Boolean equation, state machine, conditional and decode logic. It also provides access to all Altera and user-defined macro-functions.
Architecture	:	Describes the behavior, data flow, and/or structure of a VHDL entity. An architecture is created with an architecture body. A single entity can have more than one architecture. Configuration declarations are used to specify which architectures to use for each entity.
ASIC	:	Application Specific Integrated Circuit. It usually includes gate array, standard cell and full custom designs.
ASSP	:	Application Specific Standard Product. An off-the-shelf device which implements a specific function.
Behavioral Synthesis	:	Where one hardware component can be re-used for more than one parallel sequential language construction.
Bit	:	A binary digit.
BST	:	Boundary Scan Testing. An architecture which offers the capability to efficiently test components on PCBs with tight leg spacing.
Byte	:	Any 8-bit unit of data.
CCB	:	Chip Configuration Byte, which is loaded onto the Chip Configuration Register (CCR) unless the device is entering programming modes, in which case the PCCB is used.
Cell	:	A logic function.
CLB	:	Configuration building blocks.
Clocked Process	:	A synchronous process where several processes are joined with the same clock.

Combinational Process	:	A process where all the input signals must be contained as a sensitivity list.
CPLD	:	Complex Programmable Logic Device.
Design Hierarchies	:	Consist of components which contain other components, VHDL code alone, or a mixture of both.
Design Library	:	A library which stores VHDL units that have already been compiled.
Design File	:	A file that contains descriptions of the logic and is compiled by the compiler.
Device Command Mode	:	The combination of FLEX device configuration and initialization.
Device Configuration	:	The process of physically loading the SRAM programming data into the FLEX device.
Device Initialization	:	The operation in which the FLEX device resets its registers, enables its I/O pins, and begins operation as a logic device.
Device User Mode	:	Normal in-circuit device operation with the FLEX device functioning as programmed.
DSP	:	Digital Signal Processor.
EAB	:	Embedded Array Blocks. Blocks which can be used to implement various memory and complex logic functions.
EDIF	:	Electronic Design Interchange Format. An industry standard format for the transmission of design files.
Electromagnetic Interferences	:	The electrical noises other than those inherent in the circuit components.
Electromigration	:	A mass transport of metal atoms leading to electrical cuts or shorts between metal lines.
EPLD	:	EPROM Programmable Logic Devices. A PLD that uses EPROM cells to internally configure the logic function. Also called Erasable Programmable Logic Device.

Fan-in	:	The number of input signals that feed all the input equations of a logic cell.
Fast Track Interconnect	:	Dedicated connection paths that span the entire width and height of a FLEX 10K device. These connection paths allow the signals to travel between all LABs in a device.
FLEX	:	Flexible Logic Element Matrix. A FPLD offering a fast track interconnect architecture.
Floor Plan	:	The physical arrangement of functions within a design relative to the other.
FPGA	:	Field Programmable Gate Array. An array of cells that is either functionally complete or universal within a connection framework of signal rooting channels.
FPLD	:	Field Programmable logic Device. It is an integrated circuit used for implementing digital hardware that allows the end-user to configure the chip to realize different designs. Configuring such a device is done using either a special programming unit or by programming "in system".
Fuse	:	A metallic interconnect point that can be electrically changed from a short circuit to an open circuit by applying an electrical current.
GAL	:	Gate Array Logic (see description of PAL).
Gated Clock	:	A clock configuration in which the output of an AND or OR gate drives a clock.
Glitch	:	A signal value pulse that occurs when a logic level changes two or more times over a short period.
GND	:	A low-level input voltage represented as low (0) logic level in binary group values.
Input/Output Cell Register	:	A register on the periphery of a FLEX 10K device or a fast input-type logic cell that is associated with an I/O pin in some FLEX devices.

Input/Output Feedback	:	Feedback from the output pin on an Altera device that allows an output pin to be used as an input pin.
Interrupt Controller	:	The module responsible for handling interrupts that are to be serviced by user-written interrupt service routines. Also called the Programmable Interrupt Controller (PIC).
Interrupt Service Register	:	The software routine that services a standard interrupt.
IOEs	:	Input Output Element. An I/O register with flexible control signals, programmable slew-rate control and an output enable per pin.
Jam Programming	:	It is vendor and platform independent interpreted language optimized for programming devices via the standard IEEE std.1149.1 JTAG interface.
JTAG	:	Joint Testing Action Group. A specification for boundary-scan testing (BST) standardized as the IEEE 1149.1.
LAB	:	Logic Array Block. A LAB is the basic building block of the Altera MAX family. Each LAB contains at least one macrocell, an I/O block and an expander product term array.
Latch	:	A level-sensitive clocked storage unit that stores a single bit of data.
Logic Analyzer	:	Hardware equipment used to analyze digital waveforms.
Logic Element	:	A basic building block of an Altera FLEX 10K. It consists of a look-up table and a programmable flip-flop to support sequential functions.
LUT	:	Look Up Table. A function generator that quickly computes any function of four variables.
Logic Synthesis	:	The way in which Boolean functions are translated and minimized into gates.
Macrocell	:	The portion of the FPGA that is smallest indivisible building block. In the Altera MAX CPLD devices, it consists of two parts: combinational logic and configurable registers.

MAX	:	Multiple Array Matrix, which is an Altera product family. It is considered to be a CPLD.
MAX + PLUS II	:	Multiple Array Matrix Programmable Logic User System II. A set of tools that allow design and implementation of custom logic circuits with Altera's MAX and FLEX devices.
Mealy Machine	:	A machine with it's outputs being a function of the present state and all the inputs.
Mega Core Function	:	Functions that are pre-verified hardware description language (HDL) design files for complex system-level functions.
Mega-function	:	Mega-functions are off-the-shelf building blocks that implement useful functions such as processors, digital signal processing (DSP) functions, bus controllers and interfaces.
Moore Machine	:	A machine with it's outputs being a function of it's present state only.
MPLD	:	Mask Programmable Logic Device.
Netlist	:	An ASCII file that describes a design. It contains the identification of function elements, inputs and outputs and connections.
Netlist Synthesis	:	A process where a netlist is derived from an abstract presentation. This abstract presentation is usually a VHDL file.
One Hot Encoding	:	A design technique used with FPGAs and CPLDs. A single flip-flop is assigned to hold a logical one representing a state, with the rest of flip-flops being held at zeros.
OPTROM	:	One-Time-Programmable Read-Only-Memory, a version of EPROM.
PAL	:	Programmable Array Logic. A relatively small FPLD containing a programmable AND plane followed by a fixed OR plane.
PCB	:	Printed Circuit Board.

PLA	:	Programmable Logic Array. A relatively small FPLD that contains two levels of programmable logic. The one is an AND plane and the other an OR plane.
PLD	:	Programmable Logic Device. The typical device class consists of PALs, PLAs, FPGAs and CPLDs.
Primitive	:	One of the basic functional blocks used to design circuits with MAX +PLUS II software. Primitives include buffers, flip-flops, latches, logical operators, ports etc.
Propagation Delay	:	The time required for any signal transition to travel between pins and/or nodes of a device.
QFP	:	Quad Flat Plack. A device package which is physically square in nature, flat in height and surface mount.
Radio Frequency Interference	:	Interference in the form of electromagnetic radiation.
Reconfigurability	:	The ability of FPLD devices to be reprogrammed for different applications.
Routing	:	A process where previously placed logic functions are interconnected.
RTL	:	Register Transfer Logic. This consists of a language which describes behavior in asynchronous and synchronous state machines, data paths, operators and registers etc.
RTL Synthesis	:	The same as Logic Synthesis, but also translates sequential language constructions into gate and flip-flops.
SFR	:	Special Function Register.
Simulation	:	The process of modeling a logical design and it's stimuli in which the simulation calculates output signal models.
Switched Capacitance	:	The product of switching activity and physical capacitance.
Synthesis	:	The way to translate VHDL, AHDL source code into hardware.
UART	:	Universal Asynchronous Receiver and Transmitter. A part of the serial I/O port.

VCC	:	A high-level input voltage represented as high (1) logic level in binary group values. It is the default active node value in AHDL.
VHDL	:	The VHSIC Hardware Description Language. VHDL is used to describe function, interconnect and modeling.
VHSIC	:	Very High Speed Integrated Circuit.
WDT	:	Watchdog Timer, an internal timer that resets the device if the software fails to operate properly.
Word	:	Any 16-bit unit of data.

Chapter 1

Introduction

With all the available processing power currently obtainable in the form of microprocessors, it is still advantageous to develop a microcontroller based system where required peripheral components are included in the package. The trade-off between the two has been one of cost verses performance. In the design undertaken in this dissertation, the microcontroller proved the more practical option with more than sufficient hardware support being provided by the inclusion of the Flex 10K CPLDs.

The use of CPLDs in new hardware designs allows the electronic designers to develop custom support processors to implement complex algorithms in hardware. Many designers previously used ASICs or discrete logic components to implement custom hardware for dedicated applications, but with the availability of the CPLD, the designer now has the ability to speedily update, modify or emulate the new design which facilitates rapid prototyping.

Rapid prototyping provides great cost saving and allows the design to remain ahead of product obsolescence. Projecting the consumer electronic market trend implies that in the future fully customized products will need to be fabricated on demand. This indicates a definite need for the inclusion of the CPLD into most dynamic new hardware designs.

In this design the CPLDs are used to provide special purpose input/output functionality required for product customization. The new design process draws upon the expertise and multiple disciplines in order to generate an effective solution.

The choice of CPLDs verses FPGAs is due to the continuous interconnect structure of CPLDs that uses continuous metal lines to provide logic-cell to logic-cell connectivity, which ultimately leads to higher speeds and smaller die sizes than other comparable FPGAs. This continuous interconnect structure eliminates the unpredictable timing associated with a segmented interconnect structure providing fast fixed delay paths between logic cells.

Altera's range of CPLDs were chosen for the following reasons:

- The Altera PLDs are, to date, the fastest and largest in the industry, offering speeds approaching those of mainstream gate arrays.
- Altera offers the most cost-effective, highest-performance programmable logic available.
- Altera works closely with EDA manufacturers to integrate the MAX+PLUS II Software with other industry-standard design entry, synthesis, and verification tools such as those provided by Cadence, Exemplar, Logic, Mentor Graphics, Synopsis, Synplicity and View logic.
- Altera devices and development tools support both Altera-supplied megacore functions and functions created by the Altera Megafunction Partners Program (AMPP^{SM}).
- Altera devices support in-system programming which makes prototyping easy during design development. [7]

1.1 CPLDs and Rapid Prototyping

A current issue presented to most microelectronic designers today is one of integrating a field-programmable logic device with an embedded microprocessor or microcontroller. The inclusion of a microcontroller is to exploit the on-chip services which help to reduce the chip count and board area of the overall design. The use of PLDs allows the developer to deal with a high level description of a sub-system's behavior rather than with cumbersome low-level details. It is an exciting new technology which provides for a solid platform to rapid prototyping since without this technology design time would increase dramatically.

A further advantage of the CPLD is the reconfigurability of the device making it possible to re-use the hardware and once again reduce development time, while maintaining application specific performance.

The migration feature supported by Altera's FLEX 10K CPLD series allows for upgrades and application specific designs using a single printed circuit board design. Altera's FLEX 10K CPLD offers migration through logic densities ranging from 10 000 gates to 250 000 gates on pin compatible devices.

The integration of the CPLD and the microcontroller seem to be an effective solution to the problem of developing a system that could rapidly be prototyped or reconfigured to suite the immediate needs of the market.

1.2 Scope of the Thesis

This thesis presents two different approaches to an integrated hardware design utilizing firstly a microcontoller with gate-level logic technology (PHASE 1) and secondly a microcontroller integrated with two of Altera's FLEX 10K CPLDs (PHASE 2). The complete design, implementation and commissioning of both systems is presented with conclusions and recommendations following each approach.

The objectives in the development and analysis of the two systems based on their different technologies are as follows:

- Present comprehensive design techniques and methodologies as required by each design.
- Analyze the flexibility of the timing constraints to be met by both systems.
- Investigate the development times associated with each design.
- To consider the effect of PLD technology on component count and the possible reduction in the fault rate associated with this newer technology.

- Investigate the ease with which each system can be customized to suite the needs of the application.
- Utilize the hardware and software tools associated with each design technology in order to synthesize, simulate and analyze the systems where possible.
- Identify shortfalls in both designs and provide recommendations to overcome shortfalls.

1.3 Thesis Organization

This thesis is divided into 5 Chapters. Chapter 2 provides the reader with background into the electronic components utilized in the phase one and two designs. The hardware selection criteria and related hardware features used in the designs are also presented to provide the reader with insight required when developing the systems. The architecture of Altera's CPLDs and the process used to implement and design the CPLD circuits are also examined in this chapter.

Chapter 3 describes the design and analysis of the PHASE 1 microcontroller based system including the gate-level components. The conclusions and recommendations on the PHASE 1 system are then presented.

Chapter 4 presents the design and analysis of the PHASE 2 system where the microcontroller is integrated with CPLD technology. Altera's megacore functions are implemented where possible. The conclusions and recommendations are then presented for PHASE 2.

Chapter 5 concludes this thesis and presents recommendations for future work.

Chapter 2

Background

2.1 The Support Hardware

2.1.1 *Microcontrollers*

Through investigation it was determined that at least 70 percent of current microcontroller-based applications are still using 8-bit microcontrollers. For this reason it was felt that use of a cost-effective 16-bit microcontroller would provide the competitive edge desired in the market.

After a comprehensive study was performed on the available 16-bit microcontrollers, it was decided to select a microcontroller supplied by a manufacturer offering a history of sound hardware support. The microcontroller which best complied with the selection criteria was Intel's 80C196KC.

2.1.1.1 *Processing Power*

The 80C196KC is a 16-bit CHMOS microcontroller designed to handle high-speed calculations and fast input/output operations. The device has a 16-bit wide Central Processing Unit (CPU) that connects to both an interrupt controller and a memory controller via a 16-bit CPU bus. An 8-bit CPU bus transfers instruction bytes from the memory controller to the instruction register in the Register Arithmetic-Logic Unit (RALU). Figure 2.1 presents a block diagram of the internal structure of the 80C196KC.

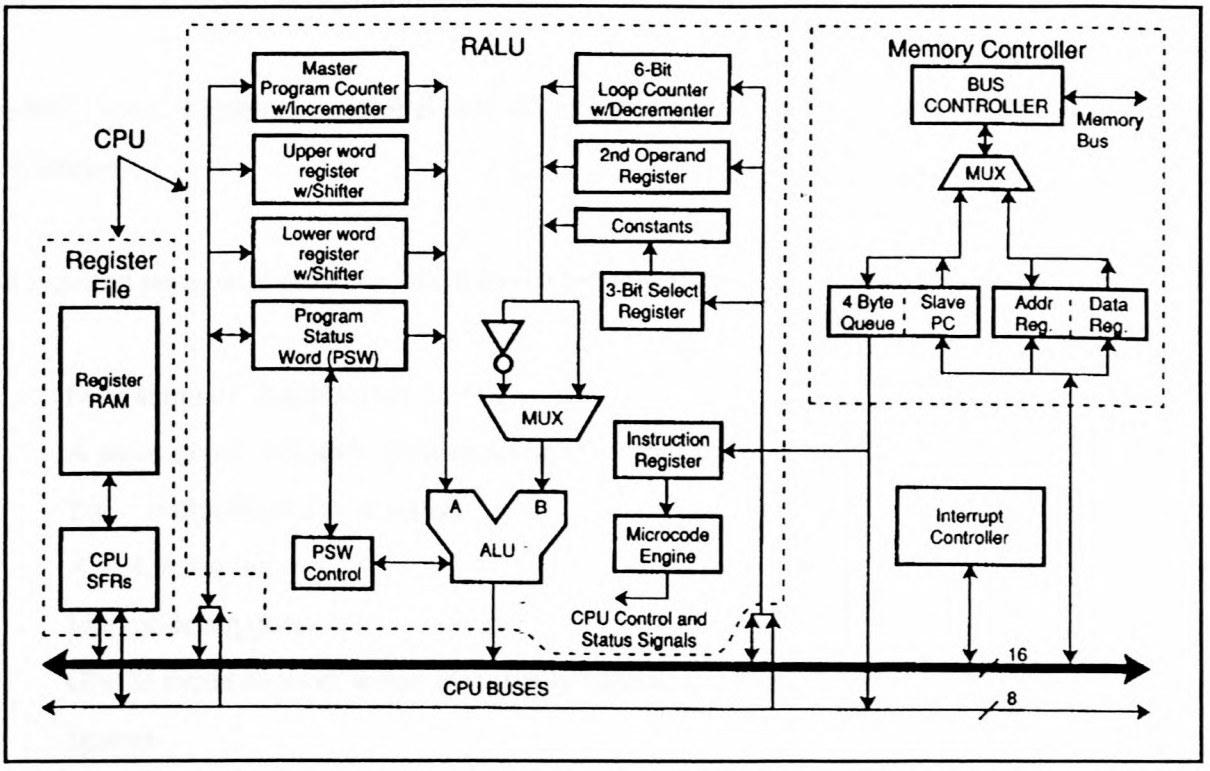


FIGURE 2.1 : A BLOCK DIAGRAM OF THE 80C196KC MICROCONTROLLER

The RALU performs most of the calculations for the microcontroller, but it does not use an accumulator. Instead, it operates directly on the lower register file, which essentially provides 256 accumulators. Because the data does not flow through a single accumulator, the 80C196KCs code executes faster and more efficiently.

For example, the 80C196KC code multiplies two 16-bit variables and stores the 32-bit result in a third variable. The 80C196KC performs this operation in one instruction because it combines a large set of general purpose registers with a 3-operand instruction format. This format allows a single instruction to specify two source registers and a separate destination register.

2.1.1.2 *Peripheral Support*

The 80C196KC's internal peripheral modules provide special functions for a variety of applications.

The required peripheral modules which led to the selection of this microcontroller are:

- Four standard input/output ports.
- A serial asynchronous/synchronous input/output port.
- Two configurable 16-bit timers.
- A watchdog timer.
- Idle mode support.
- ONCE mode support which electrically isolates the microcontroller form the external system.
- High-speed input/output (HSIO) unit.

The design implementation of these modules are presented in Chapters 3 and 4.

2.1.1.3 *Memory Support*

The addressable memory space of the 80C196KC extends to 64 kilo bytes. An INST pin on the microcontroller can be used to facilitate the overlap of ROM and RAM. This allows for an increase in the addressable memory space. Figure 2.2 shows the major memory partitions.

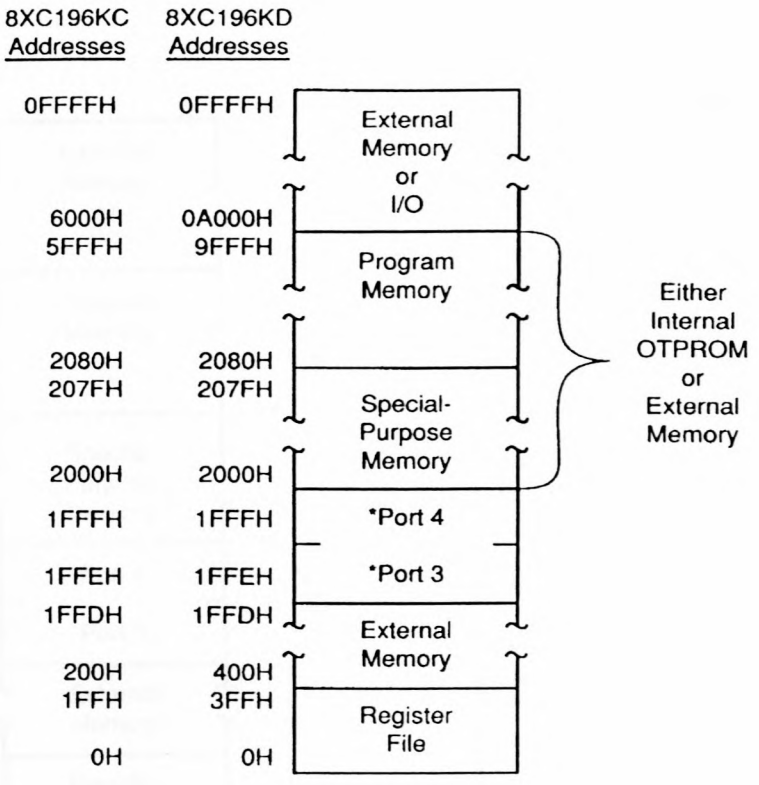


FIGURE 2.2 : MAJOR MEMORY PARTITIONS

The microcontroller can address either internal or external Read-Only Memory (ROM). The one-time-programmable-read-only-memory (OTPROM) located on the microcontroller stretches from 2000 Hex to 5FFF Hex. The 80C196KC is thus internally limited to an OTPROM of 16 k byte. The microcontroller allows for the selection of internal or external memory mapping. An external access pin (EA#) on the 80C196KC governs the selection of the internal or external memory access. Program memory is located upwards from address 2080H. Special purpose memory is located at addresses 2000H - 2070FH. It contains several reserved memory locations as presented in Figure 2.3a.

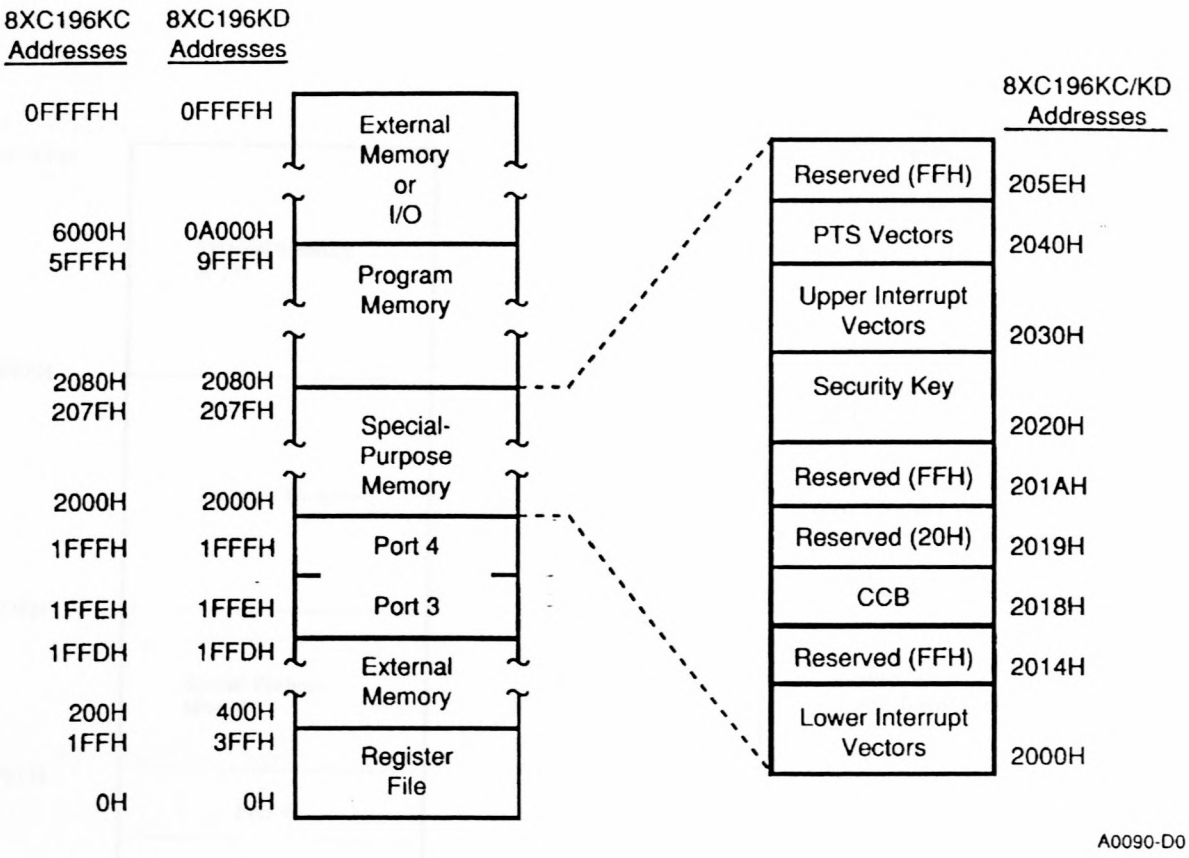


FIGURE 2.3a : RESERVED MEMORY LOCATIONS

The register file is divided into upper and lower register files as illustrated in Figure 2.3b.

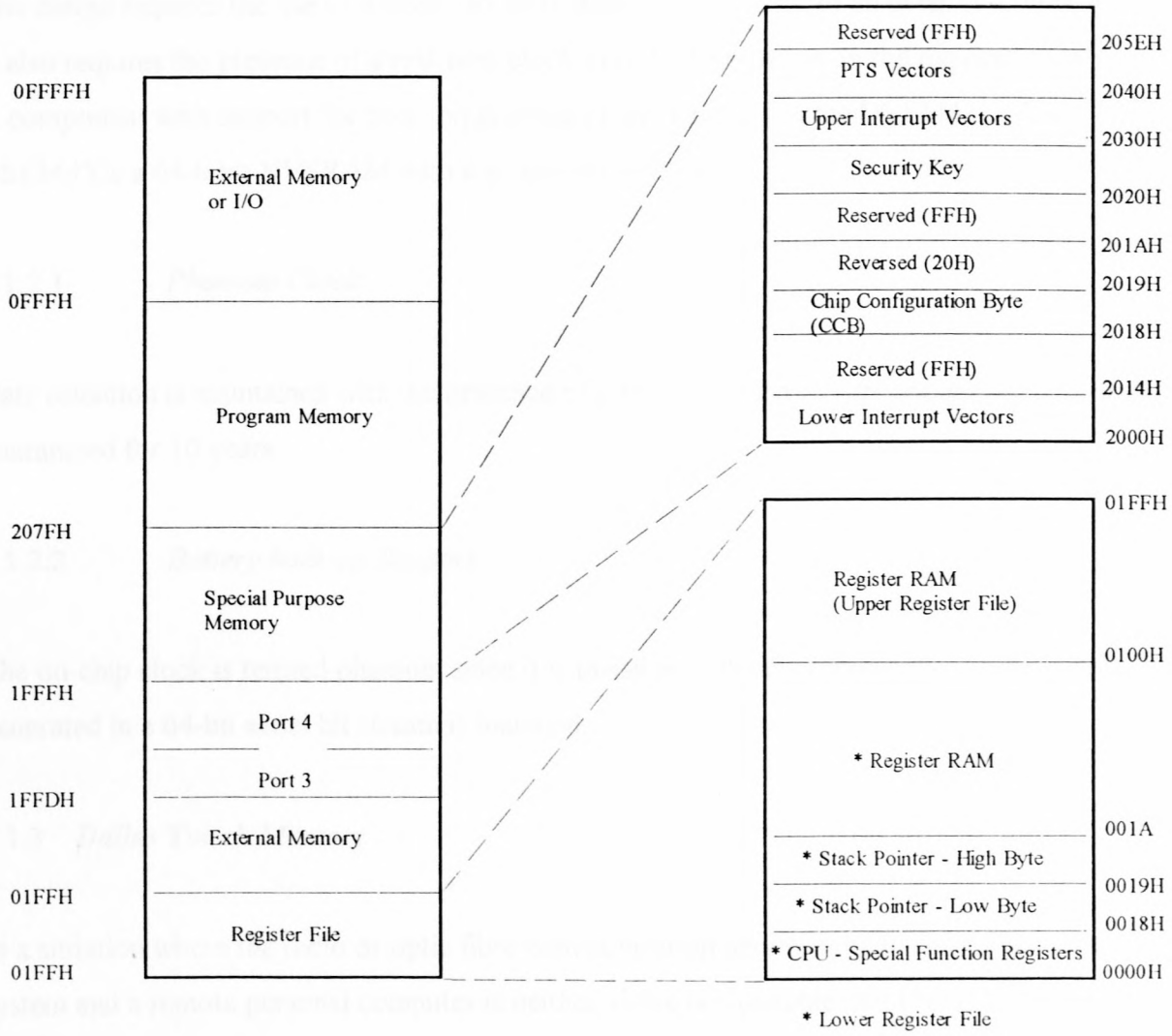


FIGURE 2.3b : REGISTER FILES OF THE 80C196KC

2.1.2 *Non-volatile ram/real-time Clock*

The design requires the use of a device to store data in the absence of an external power source. It also requires the presence of a real-time clock in order to time-stamp the required data. A component with support for both requirements was identified as the DS1244Y. The DS1244Yis a 64-k bit NVSRAM with a phantom clock.

2.1.2.1 *Phantom Clock*

Data retention is maintained with the presence of a self-contained lithium energy source which is guaranteed for 10 years.

2.1.2.2 *Battery back-up Support*

The on-chip clock is termed phantom since it is invisible to the microcontroller unless a pattern generated in a 64-bit serial bit stream is matched.

2.1.3 *Dallas Touch Memory*

In a situation where the radio or optic fibre communication between the microcontroller-based system and a remote personal computer is neither viable nor possible, the Dallas Touch Memory is utilized as a mobile data transfer mechanism.

This implies that in the event of a communication failure, all the data transferred to and from the system via the radio or optic-fibre link, can also be transferred via the mobile touch memory tags.

2.1.3.1 *Mobility*

The touch memory is a chip-based data carrier with a diameter of 16mm and a thickness of 6mm. The physical shape of the device is illustrated in Figure 2.4.

FIGURE 2.4 : DALLAS TOUCH MEMORY MODULE

Due to the small size and weight of the container, it is effortlessly transported by the user.

2.1.3.2 *Durability*

The touch memory is housed in a stainless steel can which is engraved with a unique registration number. The stainless steel can has been tested to withstand harsh environments with operating temperatures ranging from -40°C to +70°C. The can is also highly resistant to dust, moisture and shock.

2.1.3.3 *Reliability*

The manufacturer offers a 10 year guarantee on the data retention of all memory offered in the Dallas Touch Memory product range. The data integrity is assured with strict read/write protocols. The touch memory offers a unique, factory lasered and tested 64-bit registration number. This number consists of an 8-bit family code, a 48-bit unique serial number and a 8-bit CRC byte.

The memory device also complies with the UL#913(4th Edit). This implies an intrinsically safe apparatus, approved under entity concept for use in Class 1, Division 1, Group A, B, C and D locations.

The manufacturers provide a guarantee that the identity code of the device is unique. The unique 64-bit lasered ROM string is sown in Figure 2.5a. A 1-wire Cyclic Redundancy Check is generated using a polynomial generator consisting of a shift register and XOR gates as shown in Figure 2.5b. The polynomial is $X^3 + X^5 + X^4 + 1$. The CRC is used to verify the integrity of data being transferred to and from the Dallas Tag.

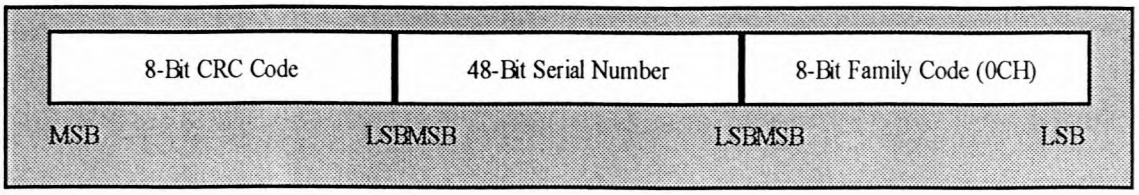


FIGURE 2.5a : 64-BIT LASERED ROM

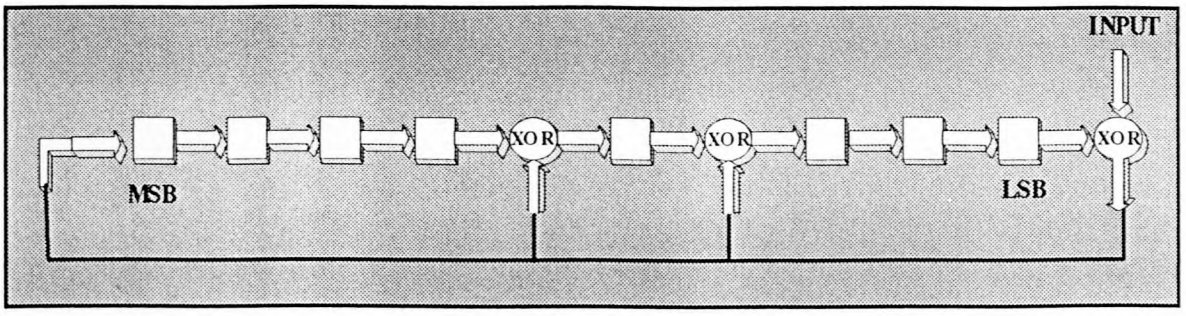


FIGURE 2.5b : 1-WIRE CRC GENERATOR

2.1.3.4 *Operation of the DS1996*

The DS1996 has three main components, the 64-bit lasered ROM, a 256-bit scratchpad and the 65536-bit SRAM. The busmaster first provides one of six ROM-function commands. 1) Read ROM [33H], 2) Match ROM [55H], 3) Search ROM [F0H], 4) Skip ROM [CCH], overdrive skip-ROM or overdrive Match-ROM [3CH or 69H]. Overdrive mode is where all subsequent communication occurs at t higher speed.

After a ROM Function Command is unsuccessfully executed, the memory functions become accessible and the master device may provide one of four memory function commands. 1) Write scratchpad [0FH], 2) Read scratchpad [AAH], Copy scratchpad [55H], Read memory [F0H]. The Memory Function Flow Chart is presented in Figure 2.6a. All the data is read and written with the least significant bit first. The other touch memory devices in the range operate similarly.

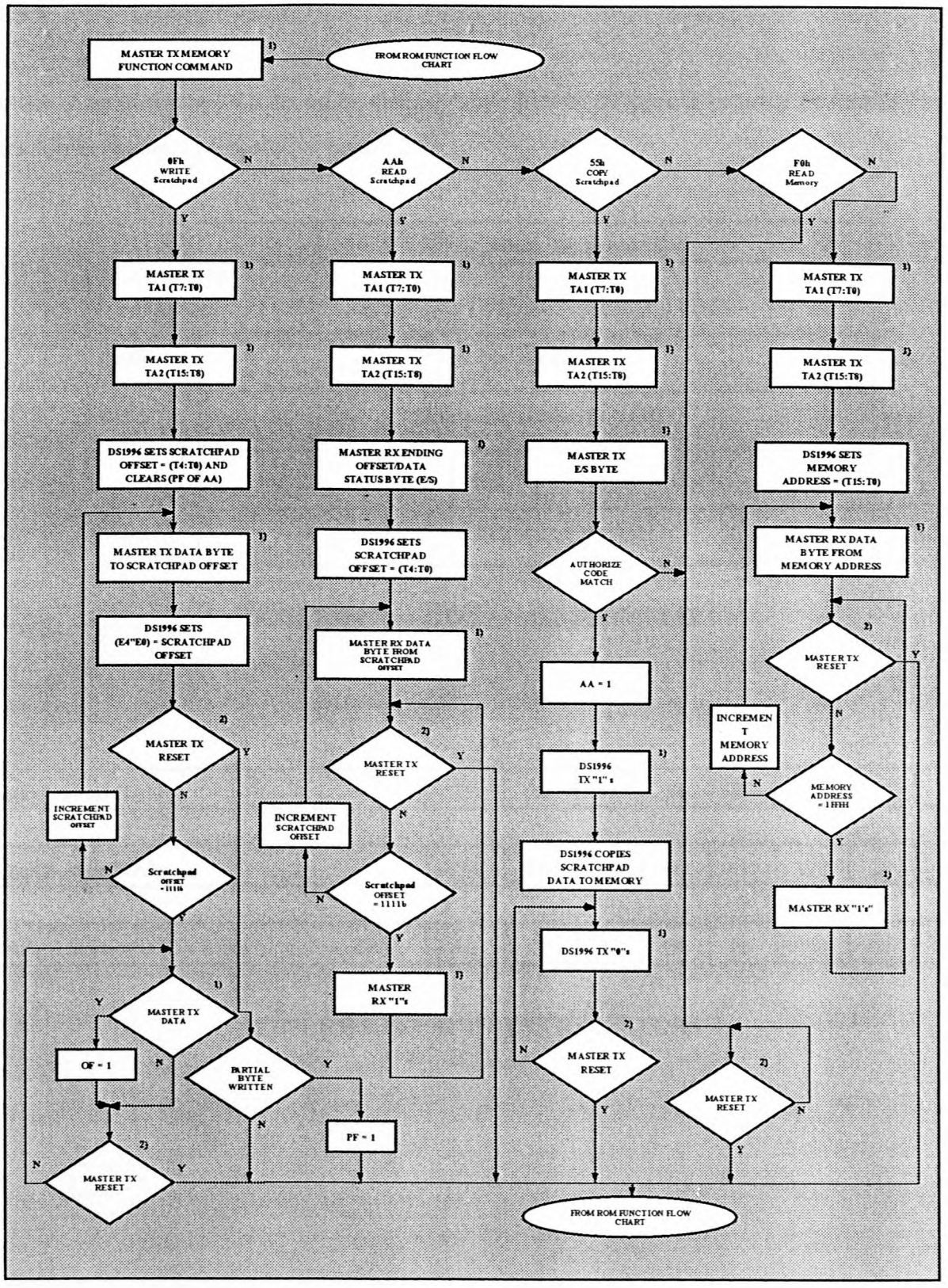


FIGURE 6.2b : MEMORY FUNCTION FLOW CHART FOR TOUCH MEMORY DEVICE

2.1.3.5 *Storage Capacity*

A block diagram of the 64-k bit touch memory is presented in Figure 2.7a, with the memory map and address registers presented in Figure 2.7b.

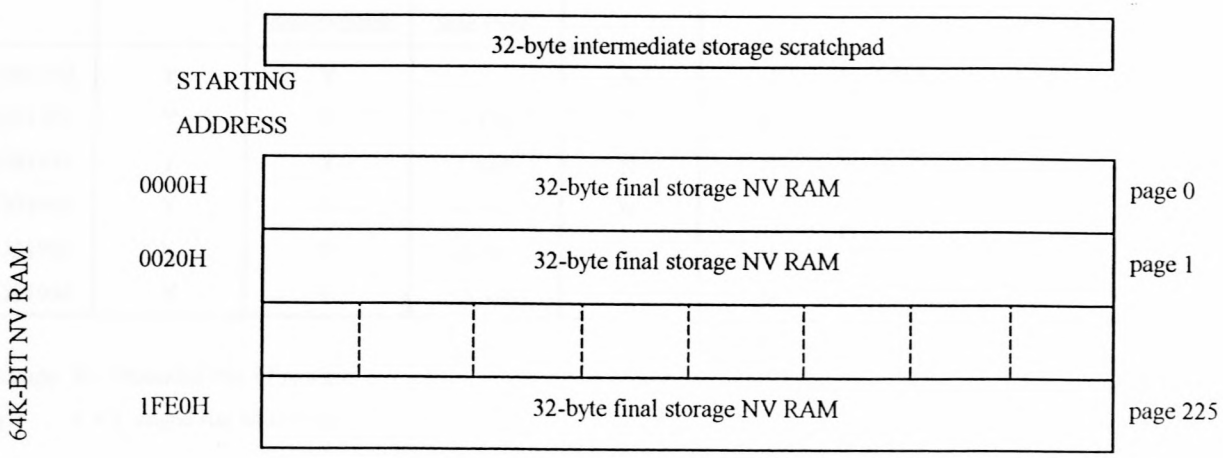


FIGURE 2.7a : DS1996 MEMORY MAP

	7	6	5	4	3	2	1	0
Target Address (TA1)	T7	T6	T5	T4	T3	T2	T1	T0
Target Address (TA2)	T15	T14	T13	T12	T11	T10	T9	T8
Ending Address with Data Status (E/S) (Read Only)	AA	OF	PF	E4	E3	E2	E1	E0

FIGURE 2.7b : ADDRESS REGISTERS

Features of the Dallas Touch Memory used in the design is presented in Table 2.1

Touch Memory Family								
	1-Wire Compatible	48-Bit Unique Identification	Read/Write Memory Size (bits)	Secure Memory	Real-Time Clock	Maximum Data Rate	Package Style	Stainless Steel
DS1990	Y	Y	0	N	N	16 kbps	R	Y
DS1991	Y	Y	1,152	Y	N	142 kbps	R/F	Y
DS1992	Y	Y	1,024	N	N	142 kbps	R/F	Y
DS1993	Y	Y	4,096	N	N	142 kbps	R/F	Y
DS1994	Y	Y	4,096	Y	Y	16 kbps	R/F	Y
DS1996	Y	Y	65,536	Y	N	142 kbps	R/F	Y

Package R = Rounded rim MicroCan
Style F = Flanged rim Microcan

TABLE 2.1 : TOUCH MEMORY FAMILY

The 64 k-bit touch memory is used for the transfer of recorded transactions. The transactions are transferred from the microcontroller-based system to the database located on the personal computer. The 4-k bit touch memory is used for system configuration and interrogation with date and time synchronization. The 1-k bit touch memory is used as a user identification tag containing system interrogated user control data.

2.1.3.6 *Parasitic Power*

The block diagram indicating the parasitic-powered circuitry is presented in Figure 2.8.

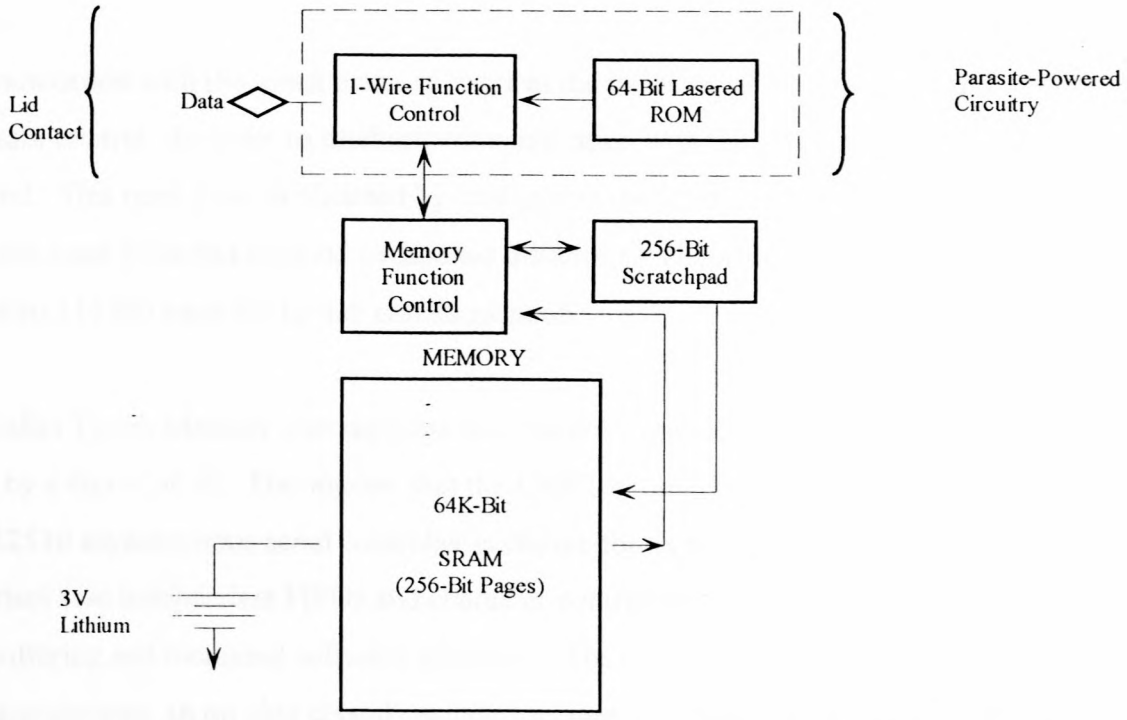


FIGURE 2.8 :INTERNAL STRUCTURE OF THE DALLAS TOUCH MEMORY

The circuitry "steals" power whenever the data line is high. The data line provides sufficient power as long as the specified timing and voltage of the parasitic power is that the lithium cell is conserved and that if the lithium cell is exhausted of energy, the ROM identity serial number can still be accessed.

2.1.3.7 *Ease of Implementation*

The touch memory tags are rugged read/write data carriers that act as localized databases that can easily be accessed with minimal hardware. The data is transferred serially via a 1-wire protocol which requires only a single data lead and a ground return.

2.1.4 *Intel 82510 Universal Synchronous Receiver and Transmitter (UART)*

The design requires the use of two UARTS. One for serial communication with a remote personal computer and the other for communication with the Dallas Touch Memory.

Communication with the touch memory requires the inclusion of an external UART with flexible baud rate control. In order to establish communication with the touch memory, a reset pulse is required. This reset pulse is obtained by configuring the band rate of the UART to 10473 band. Once the reset pulse has initiated a response from the touch memory, the band rate must then be altered to 115200 baud for further communication.

The Dallas Touch Memory also supports an 'overdrive' mode which increases the communication speed by a factor of 10. This implies that the UART chosen should support the new speeds. The Intel 82510 asynchronous serial controller is chosen for its intelligence and versatility. The UART comprises two independent FIFOs and character control recognition (CCR) for the purpose of data buffering and increased software efficiency. The controller also has two baud rate generators/timers, an on-chip crystal oscillator and seven programmable I/O pins. A block diagram of the UART is presented in Figure 3.14.

2.1.4.1 *Register Support*

The Intel 82510 has a set of registers which can be configured to appear identical to those of the 16450 UART. This feature provides complete software compatibility, with the IBM personal computer environment.

2.1.4.2 *Reduced Power Consumption*

Both the Intel 80C196KC microcontroller and the Intel 82510 UART support the "sleep" and "idle" modes which reduce unnecessary power consumption.

2.1.5 *Programmable Logic Devices (PLDs)*

The address decoding requirements for the phase 1 hardware is implemented with the use of GALs (Gate Array Logics), rather than discrete logic components. The 22V10-GAL was selected for it's logic capacity and access time in order to comply with the design requirements of the hardware. The minimum access time allowed for is 7,5 ns as reflected in the hardware timing specification presented in Appendix A.

Due to the flexibility required from the hardware, a state machine is coded into the GAL in order to facilitate multiple wait-states for a range of access times.

2.1.5.1 *Configurability*

A state machine is used to implement wait-states for the Intel 196 microcontroller in a 22V10 GAL. The reason for this being that the performance and time determinism are considerably better in state machines implemented in gates and flip-flops than in a CPU. The two basic types of machines considered were the Moore and Mealy. The Mealy machine's outputs are a function of the present state and all of the inputs whereas the Moore machine's outputs are a function of the present state only. The Mealy machine then changes it's outputs immediately when the inputs change whereas the Moore machine has to change state first. In the situation where the state machine is used to implement a wait-state system, the Moore machine is the relevant choice since it would wait for a clock cycle before the output values can be changed. This is illustrated in Figure 2.9.

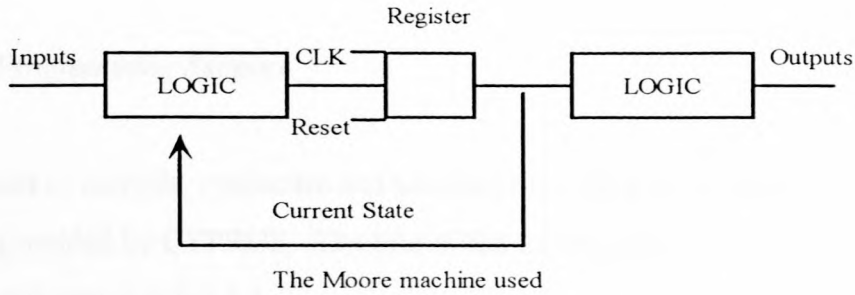


FIGURE 2.9 : MOORE MACHINE DESIGN

The state machine is implemented in the VHDL language then compiled and tested using available synthesis and simulation tools. Since the state coding does not affect the behavior of the state machines, it does not have to be described in the VDHL code. The synthesis tool usually includes a state machine optimizer which is used to optimize for desired performance and area. Due to the fact that the synthesis tool originally used in the development of the phase 1 system did not have a state machine optimizer, the state coding was included in the VHDL code.

It is accepted that both the Moore and Mealy machines can have signal spikes on the outputs but with the state decoding assigned in the VHDL code, the outputs are guaranteed to be spike-free. This is due to the fact that the outputs are generated directly from the state flip-flops. The output used as a wait-state signal is a clocked output and hence also yields a spike-free signal via the extra D-type flip-flop. This is illustrated in Figure 2.10.

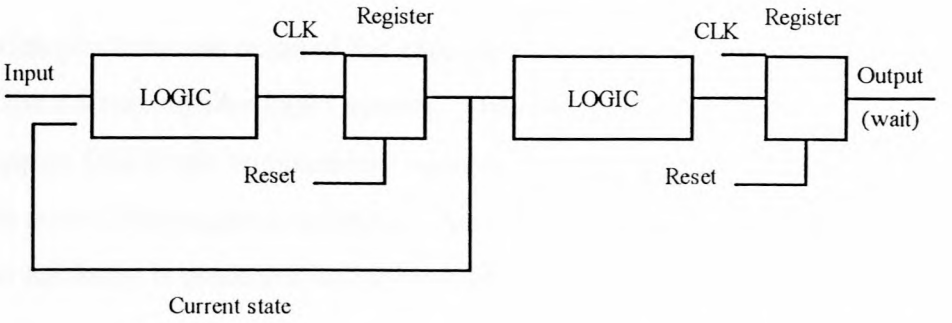


FIGURE 2.10 : MOORE MACHINE WITH CLOCKED OUTPUT

A synchronous state machine was used rather than an asynchronous machine in order to produce the required results.

2.1.5.2 *Programming Support*

The software used to compile, synthesize and simulate the VHDL code used to program the 22V10 GAL is provided by CYPRUS. The title of the software package is WARP and the version used in this design is ver 3.3

2.1.5.3 *Reliability*

The use of PLD technology introduces an increased sense of reliability due to reduced component count, internal connectivity and sophisticated software analysis support.

2.1.5.4 *Cost-effectiveness*

The cost of a single PLD is far less than the cost of the equivalent gate-level hardware required to accomplish the same task. Design time is also greatly reduced due to the reconfigurable nature of the PLD.

2.1.6 ***Optic Fiber Communication***

A communication platform was required for data communication between a remote personal computer and the microcontroller-based system. The communication hardware had to be designed to support RS232 pin compatibility in order to operate off the standard serial communication port of the personal computer. A block diagram illustrating the design of the communication hardware is presented in Figure 2.11.

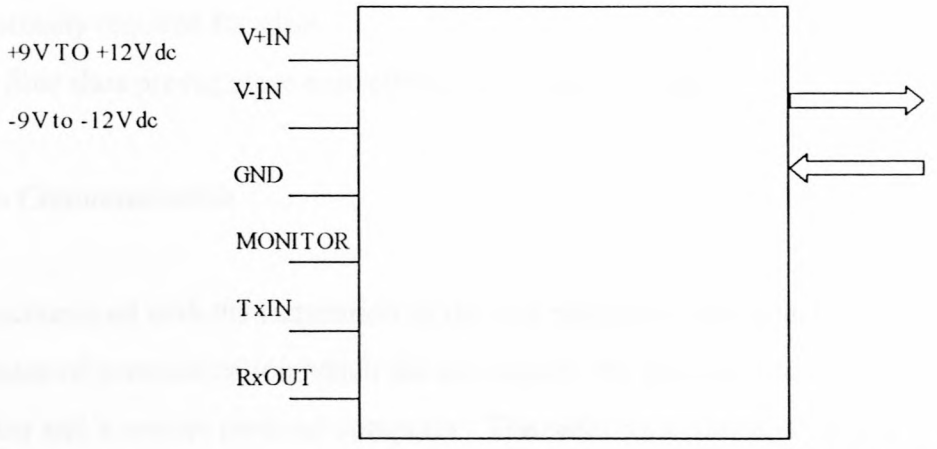


FIGURE 2.11 : OPTIC FIBER COMMUNICATION HARDWARE

The criteria which led to the selection of the communication are as follows:

2.1.6.1 *Reliability*

The communication system has to be reliable over a distance of up to 100 m and the system has to support a band rate of 19 200. Due to the nature of the project, the system needs to be immune to electrical noise found in an industrial environment.

2.1.6.2 *Safety*

The project is developed for use in the fuel industry and a great deal of safety precautions need to be considered when designing such a system. The optic fiber system provides full optic isolation between the remote personal computer and the microcontroller-based system. The system also offers intrinsic safety in regions of frequent lighting strikes.

2.1.6.3 *Cost-effectiveness*

The nature of the project required the use of a low cost polymer fiber to be used instead of glass. The polymer fiber allows for terminations to be performed without the need for expensive equipment normally required for glass.
The polymer fiber thus proves more cost-effective to install or repair.

2.1.7 ***Radio Communication***

A problem encountered with the installation of the fuel management system brought about the need for a means of communication which did not require any physical connection between the microcontroller and a remote personal computer. The radio communication system incorporates a communication module with a prepaid frequency band. This module is manufactured by Radiometrix. Additional circuitry had to be designed around the module in order to make the complete radio transceiver RS232 compatible.

The transceiver unit operates at the UHF frequency of 433 MHz and supports half duplex data transmission at speeds up to 40-k/bits per second. The range obtainable from the unit is 120 meter in open ground. A block diagram of a fully buffered CMOS interface is presented in Figure 2.12.

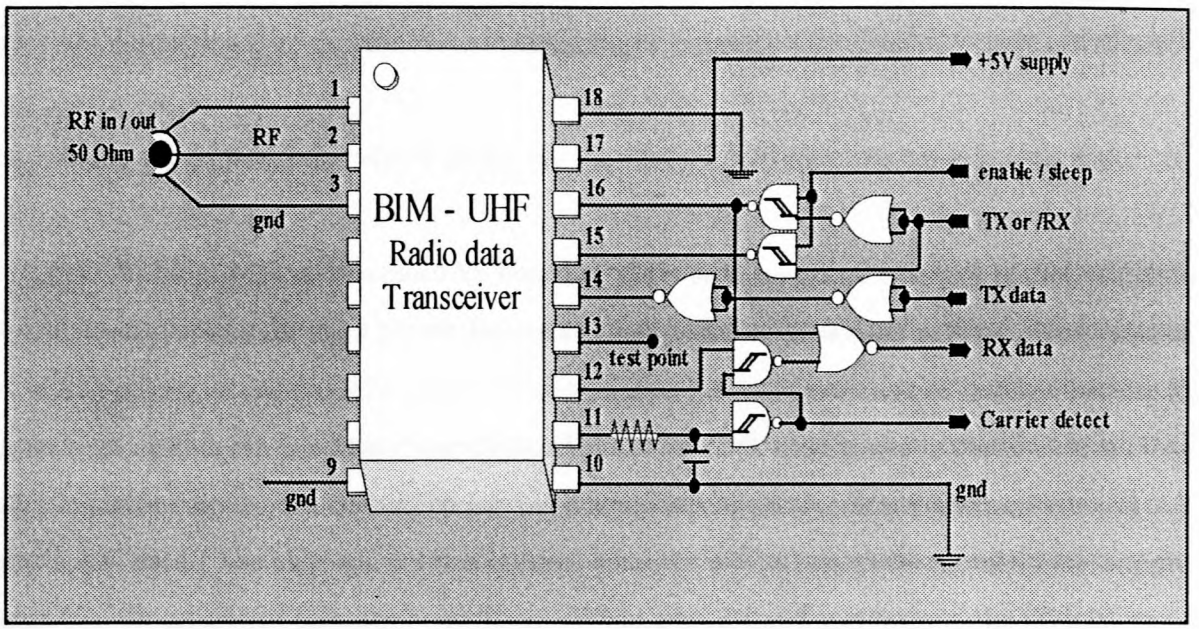


FIGURE 2.12 : RADIO COMMUNICATION HARDWARE

2.1.7.1 *Reliability*

In order to improve the bit error rate performance on data sent, a 50:50 mark:space average is implemented. This is accomplished by transmitting each byte twice. The first transmission sends the actual byte and the second transmission sends it's logical compliment. The decoder thus determines the integrity of each byte received and on a parity failure selects the subsequent inverted odd byte.

2.1.7.2 *Ease of Implementation*

The modular nature of the design facilitates component replacement rather than on-site fault diagnosis. The RS232 pin compatible design supports the plug-in replacement of the optic fibre communication module with the radio communication module. The wireless nature of the radio transceiver eliminates the need for cable installation or repairs.

2.1.8 ***Microcontroller Supervisory Support***

A common problem with microcontroller based systems with on-board memory is that of data-loss or data-corruption during a power failure or fluctuation in the supply voltage. The memory used in this project is non-volatile battery backed with a 10 year guarantee of on-chip battery life. The package used is the Dallas semiconductor DS1244Y 32 K byte non-volatile SRAM with an on-chip phantom clock. An embedded lithium energy cell maintains calender operation and retains RAM data. The chip has internal control circuitry which constantly monitors the supply voltage V_{CC} for an out-of-tolerance condition. When such a condition occurs, the lithium energy source is automatically switched on and write protection is unconditionally enabled to prevent corrupted data in both the memory and the real-time clock.

The DS1244Y provides full functional capability for V_{CC} greater than a set threshold voltage of 4.25 Volts and write protects for any voltage below 4.25 Volts. When the supply voltage decays, the RAM automatically write protects itself with all the inputs becoming "don't cares" and the outputs reverting to a high impedance state.

As V_{CC} falls below approximately 3.0 Volts, the power switching circuit connects external V_{CC} to the RAM and disconnects the lithium energy source. Normal RAM operation once again resumes after V_{CC} exceeds 4.5 Volts.

One of the issues which is however not addressed by the secure write protection strategy of the NVSRAM is that there is a strong likelihood that data is currently being processed during a power failure. This implies that the now fluctuating bus states are resulting in erroneous data or addresses being passed in the system.

An example of this would be when the supply voltage (V_{CC}) has fallen below the allowable 10% of the NVSRAM phantom clock and time has already elapsed for performing any system housekeeping functions such as storing off data and storing the state of the microprocessor. An essential requirement would be to save the data before write-protecting the memory.

This could only be done if the impending power failure could be detected well in advance of the 10% "shut-off-point". This early detection would allow enough time for the essential housekeeping functions to be performed. The system implemented in this project is considered sufficiently reliable for production systems where data integrity is a high priority.

2.1.8.1 *Power Monitoring Systems*

Two separate power monitoring systems are implemented together in order to provide a "back-up" in the event of any one of the two systems failing. The two systems chosen were selected from different manufacturers in order to avoid the occurrence of common faults.

The first power monitoring system uses the DS1233 5% and the DS1233 10% 5 Volt reset semiconductor chips manufactured by Dallas Semiconductor. The two ICs operate in sequence during a power failure. The first IC withdraws the signals from the busses then the second IC issues a microcontroller reset.

The second power monitoring system uses the MAX695 chip manufactured by MAXIM. This chip takes control of the chip select signal to the NVSRAM during a power failure to avoid unwanted chip access. The MAX695 also withdraws the signals from the busses and then issues a microcontroller reset on power failure.

The full implementation of both systems is presented in Chapters 3 and 4. The first system is used in conjunction with the DS1244Y in the PHASE 1 design, while in PHASE 2 the second system is implemented as a back-up for additional protection against data corruption.

2.1.9 *CPLDs and FPGAs*

In the production and development of electronic systems today, there is a drive to increase the use of high density PLDs because of the current similarities between these PLDs application specific integrated circuit (ASICs) and application specific standard products (ASSPs) [7]. The factors influencing this increased use of high density PLDs are:

- high integration
- high density
- high performance
- low cost
- short time-to-market
- design flexibility

Programmable logic includes all of the digital logic circuits configured by the end-user, which includes the simple, low-density. Generic Array Logic (GALs), Programmable Array Logic (PALs), Field Programmable Gate Arrays (FPGAs) and Complex PLDs (CPLDs). Figure 2.13 illustrates the CMOS logic product range offered by Altera [7].

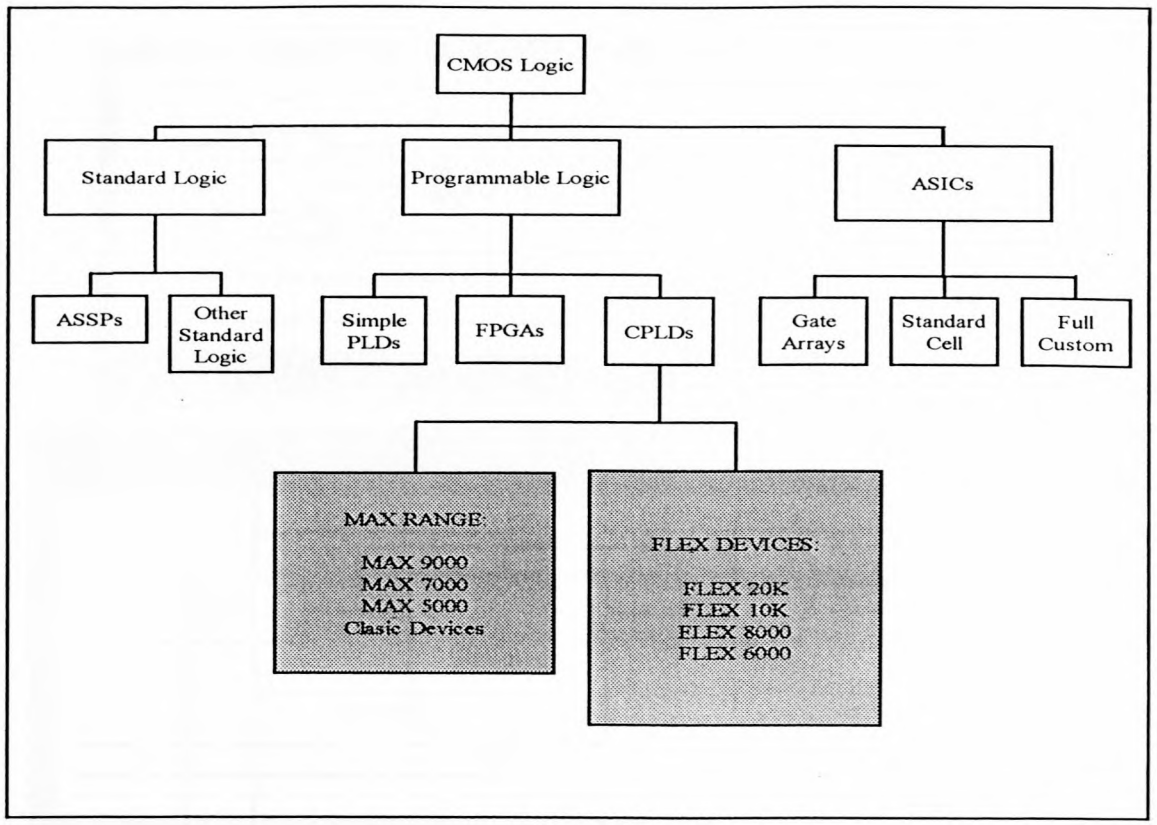


FIGURE 2.13 : ALTERA'S CMOS LOGIC PRODUCT RANGE

The two prominent high density logic devices manufactured today are the FPGAs and the CPLDs. There exists great controversy between the use of each of these components with manufacturers choosing to specialize in the one or the other. The CPLDs and FPGAs have different interconnect structures. The segmented interconnect structure of the FPGA used multiple metal lines of ranging lengths, connected by pass transistors or anti-fuses, to connect logic cells.

The CPLD uses a continuous interconnect structure with continuous metal leads to higher speeds and smaller die sizes than comparable FPGAs. A further feature in support of the CPLD is that the continuous interconnect structure eliminates the unpredictable timing associated with a segmented interconnect structure and provides fast, fixed delay paths between logic cells. Figure 2.14 illustrates the architectural structures and differences between the CPLDs and FPGAs.

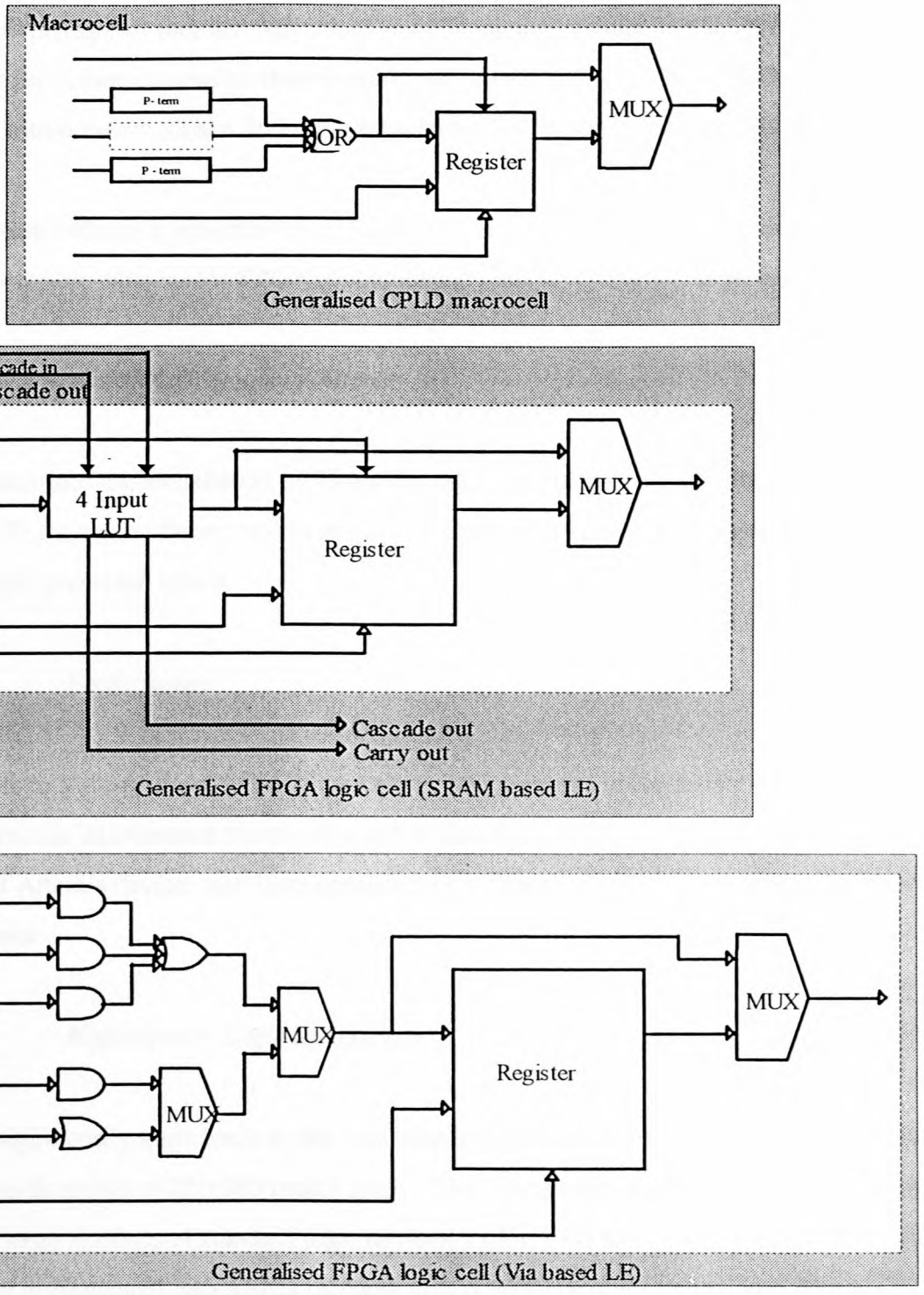


FIGURE 2.14 : ARCHITECTURAL STRUCTURES OF CPLDs AND FPGAs

Other differences between CPLDs and FPGAs include the fact that while CPLDs are gate intensive implying that they are well suited to wide input functions, FPGAs are register intensive, but where input functions wider than 4 would require extra levels. The CPLD is not register intensive with between 32 and 560 registers where the FPGA has between 282 to +5000 registers.

An important difference between CPLDs and FPGAs is that currently the latter is non-volatile, but re-programmable whereas the FPGA is volatile and must be re-booted at power-up.

2.1.10 ***Flex Series CPLD Range of Altera***

The selection and implementation of Altera's products is attributed partly to the quote that "Altera's PLDs are the fastest and largest in the country". [7] Further reasons supporting this selection are presented below.

2.1.10.1 *Performance*

Altera utilizes state-of-the-art CMOS processes, which results in the fastest possible performance. The continuous interconnect structures result in fast and consistent signal delays in the device. Certain of Altera's devices also have specialized on-chip circuitry which further enhance the performance.

2.1.10.2 *High-density Logic Integration*

Altera's high-density logic leads to the reduction in board space and cost. Altera's devices range in densities from 300 to 250 000 usable gates. The logic densities allow the devices to easily integrate large numbers of standard logic devices, PLDs, FPGAs or custom devices. Applications previously implemented into ASICs can now just as easily be implemented into Altera's FLEX devices.

2.1.10.3 *Cost-effectiveness*

"Altera offers the most cost-effective, highest performance programmable logic available". [7] It is also important to note that Altera's PLDs are similar in cost to equivalent gate array logic components.

2.1.10.4 *Development Cycle*

The shorter development cycles can directly be attributed to the easy-to-use MAX+PLUS II software. With Altera's software, the design entry, processing, verification and device programming require only a few hours to complete. The MAX+PLUS II development cycle is presented in Figure 2.15

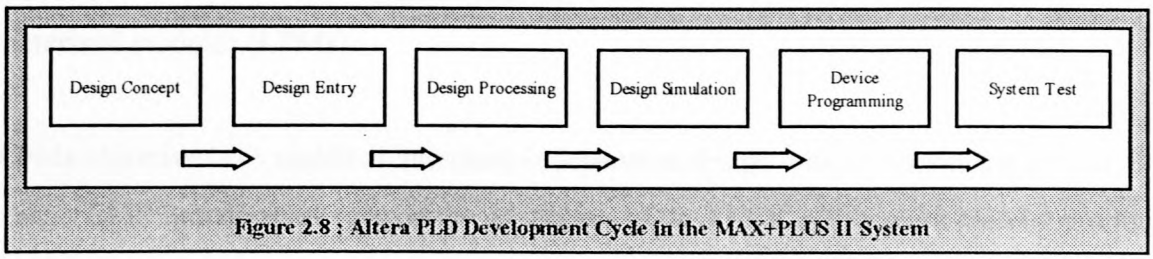


Figure 2.8 : Altera PLD Development Cycle in the MAX+PLUS II System

FIGURE 2.15 : MAX+PLUS II DEVELOPMENT CYCLE

Altera has supported the development of direct integration of most EDA manufacturers software with its own MAX+PLUS II. Altera has developed its MAX+PLUS II software with support for industry-standard design entry, synthesis and verification tools. This project makes use of the synopsis design verification tool, which contributed to the choice of MAX+PLUS II software as the possible solution.

2.1.10.5 *Mega-function Support*

Altera provides MegaCore functions and supports the Altera Mega-function Partners Program (AMPP^{SM}) mega-functions. Mega-functions are off-the-shelf building blocks that implement useful functions such as processors, Digital Signal Processing (DSP) functions, bus controllers, and interfaces. [7] The mega-functions provide a high degree of flexibility and performance which is not attainable in fixed-function devices. These functions can be targeted to various applications required by the design.

– *LPM-Megafunctions*

Altera's FLEX10K devices have created a paradigm shift in design methodology. Designers are moving away from schematic-based design techniques and are turning to modern design techniques that use hardware description languages (HDLs), megafunctions and libraries of parameterized modules (LPMs).

The LPMs objective is to enable architecture-independent design without sacrificing silicon efficiency. LMP functions take maximum advantage of the FLEX architecture thereby producing high performance functionality. For instance a multiplier function implemented in a FLEX device out-performs a multiplier function implemented in competing devices [26].

For performance optimization, designers can use the LPM-latency parameter to create a pipelined multiplier that is suited to the FLEX architecture, utilizing the many chains. Competing vendors provide significantly less, if any support for the LPM. For example, the XBLOCK methodology used by Xilinx, does not support signed multipliers which are vital for certain application the LPM-counter and the LPM-MUX megacore-functions.

The Altera MegaCore functions are pre-verified Hardware Description Language (HDL) design files for complex system-level functions. [7]

2.1.10.6 *JAM Programming and Test Program*

The JAM programming solution consists of two software components. The first being the JAM composer and the second the JAM player. Figure 2.16 describes the basic JAM flow.

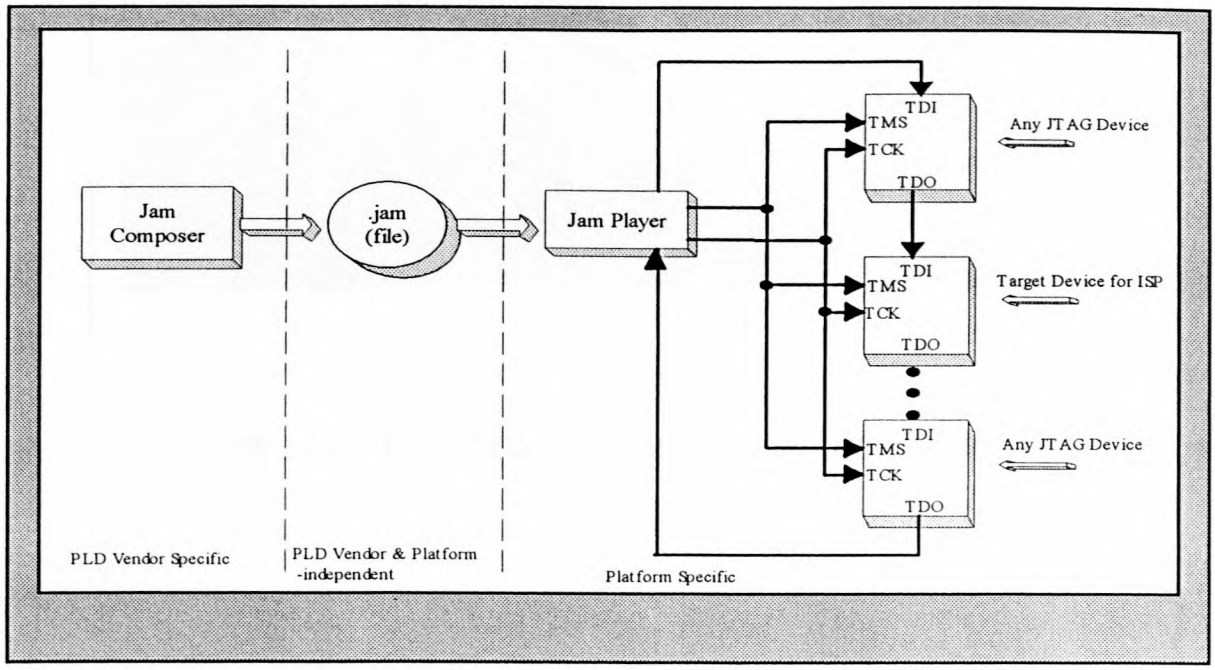


FIGURE 2.16 : FLOW DIAGRAM OF ALTERA's JAM SYSTEM

The JAM composer writes the JAM file (*.jam) that contains the user data and programming algorithm which is required to program the design into a device. The JAM player interprets the JAM file and manages the JTAG (Joint Test Action Group) port to program the device. The JAM instruction set includes JTAG-based instructions as well as algorithmic instructions.

Most Altera devices comply with the JTAG specification IEEE 1149.1 std by providing BST capability for input, output and dedicated configuration pins. The BST architecture can test pin connections without the use of physical test probes and capture functional data while the device is operating normally. [32] In order to perform the boundary test, the data has to be serially forced into the boundary scan cells. The captured data is then serially shifted out and externally compared to expected results. This technique is illustrated in Figure 2.17a and 2.17b.

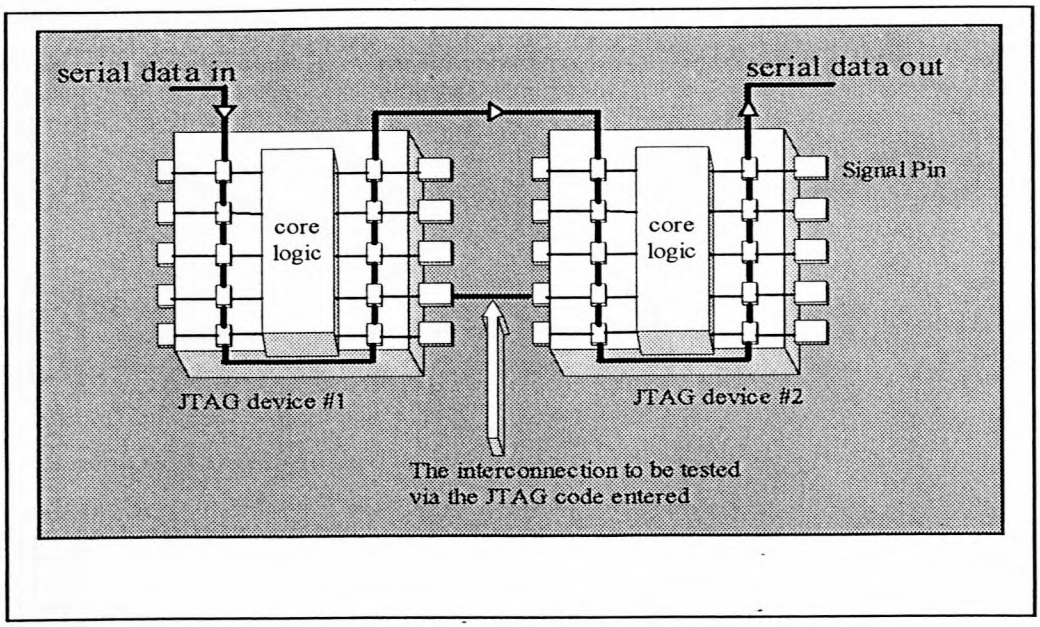


FIGURE 2.17a : IEEE STD. 1149.1 BOUNDARY-SCAN TESTING

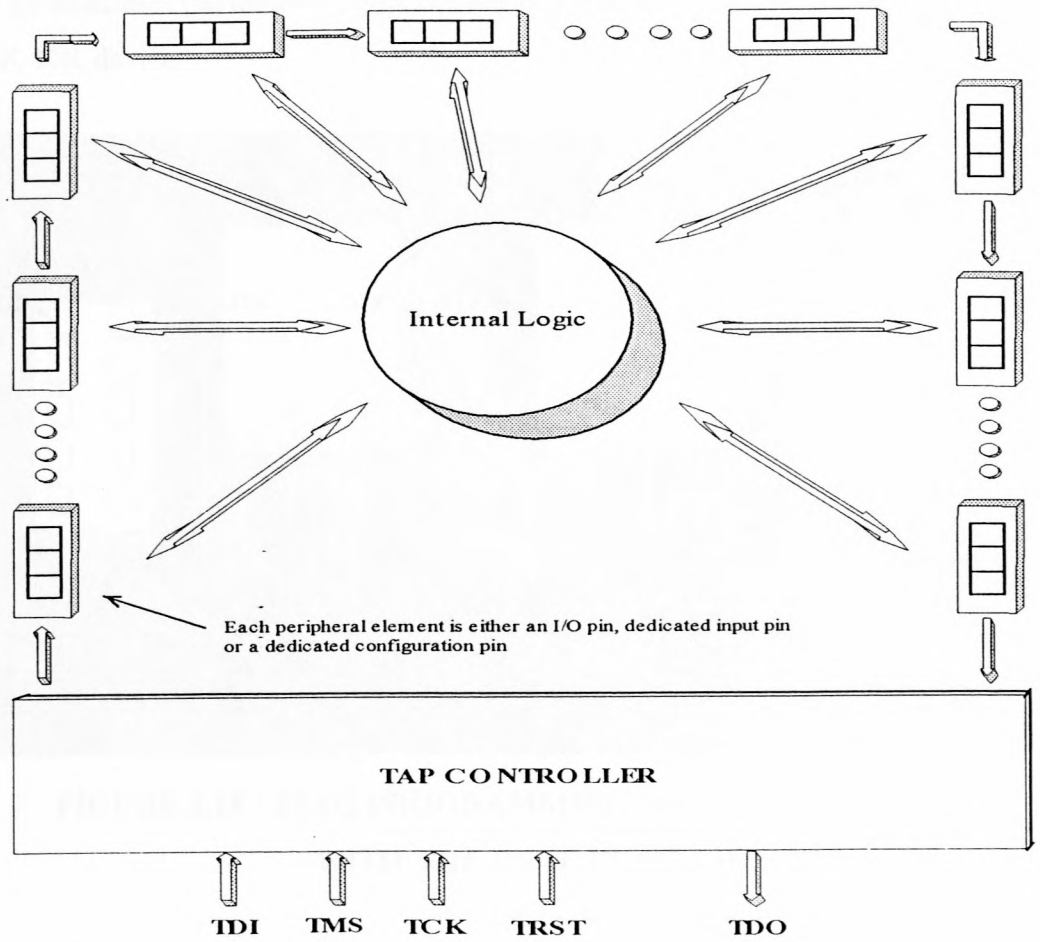


FIGURE 2.17b : BOUNDARY SCAN TESTING LAYOUT

The BST technology can be used for in-circuit reconfiguration in the FLEX 10K devices used in this project.

Figure 2.18 illustrates the hardware connections required to implement the BST technology for the FLEX 10K device.

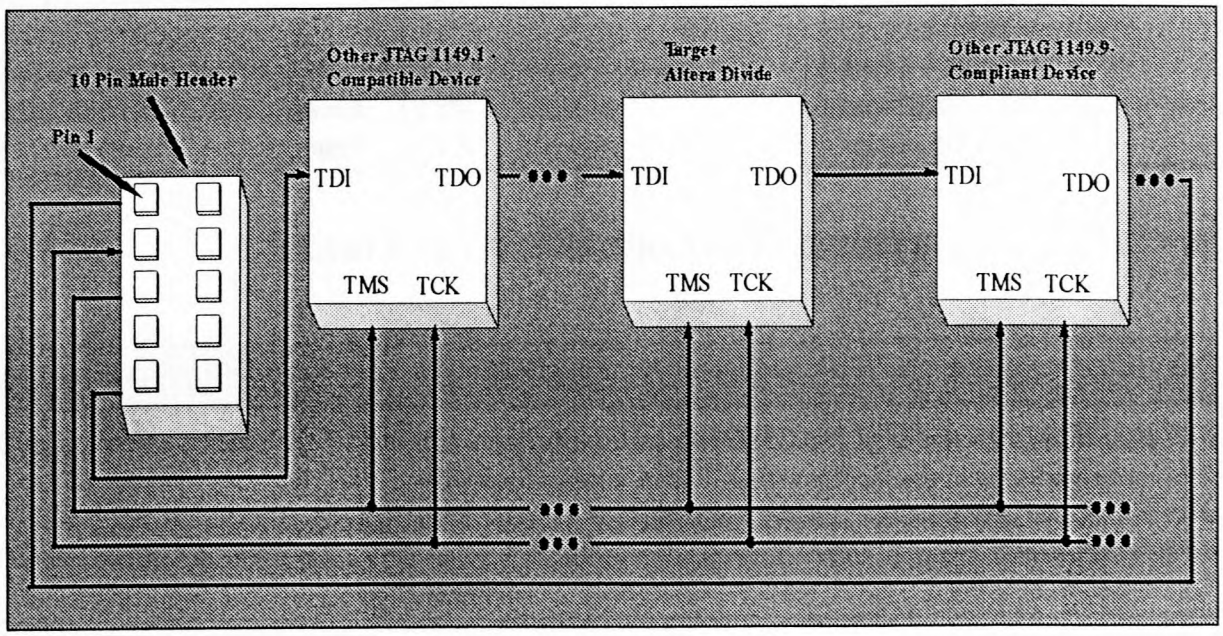


FIGURE 2.18 : JTAG PROGRAMMING AND CONFIGURATION WITH THE BYTE BLASTER

The BST mode requires the use of 5 specific pins. They are the TDI (Test Data Input), TDO (Test Data Output), TMS (Test Mode Select), TCK (Test Clock Input) and the TRST (Test Reset Input), where the TRST pin is optional. A description on the implementation of this hardware is described in Chapter 4, section 4.1.5.

2.1.10.7 *In-circuit Configurability*

The Altera FLEX 10K device supports in-circuit re-configurability. The FLEX 10K device allows various configuration schemes to be selected on power-up. These are presented in Table 2.2 with their typical uses presented in Table 2.3.

– *Configuration Methods*

The FLEX 10K architecture uses SRAM cells to store the configuration data for the device. The SRAM cells have to be reloaded each time the circuit powers up and begins operation. The devices configuration mode is entered, followed by the device initialization mode placing the device into user mode where the device functions as programmed. The FLEX 10K device can be either configured on power-up or during operation by allowing an external device to take control of the dedicated pins assigned to configuring the device.

Initialization of the FLEX 8000 device can be controlled by the internal oscillator in the device or by an external clock signal, whereas the FLEX 10K device initialization can only be controlled by an external clock signal. This clock is provided by the EPROM or intelligent controller. Once the device is reconfigured in-circuit, normal user-mode operation can resume. The process from configuration to user mode takes less than 320 mseconds. A timing diagram illustrating the configuration cycle of the FLEX 10K device is presented in Figure 2.19.

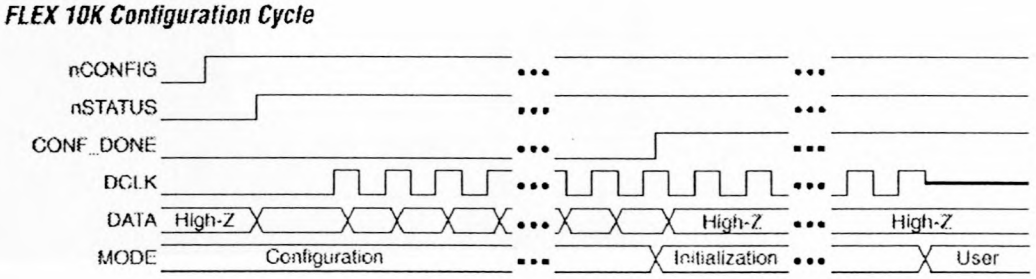


FIGURE 2.19 : TIMING DIAGRAM FOR CONFIGURATION OF FLEX 10K DEVICE[41]

The EPC2, EPC1 or EPC1441 EPROM devices store the configuration data in its array and serially clocks the data out with an internal oscillator. On the EPROM, the OE, NCS and DCLK pins supply the control signals for the address counter and output tri-state buffer. The passive serial configuration methods used in the project also allows for the configuration of a single FLEX 10K device or multiple FLEX 10K devices. Figure 2.20 shows the pin connections for the EPROM configuration of both single and multiple FLEX 10K devices.

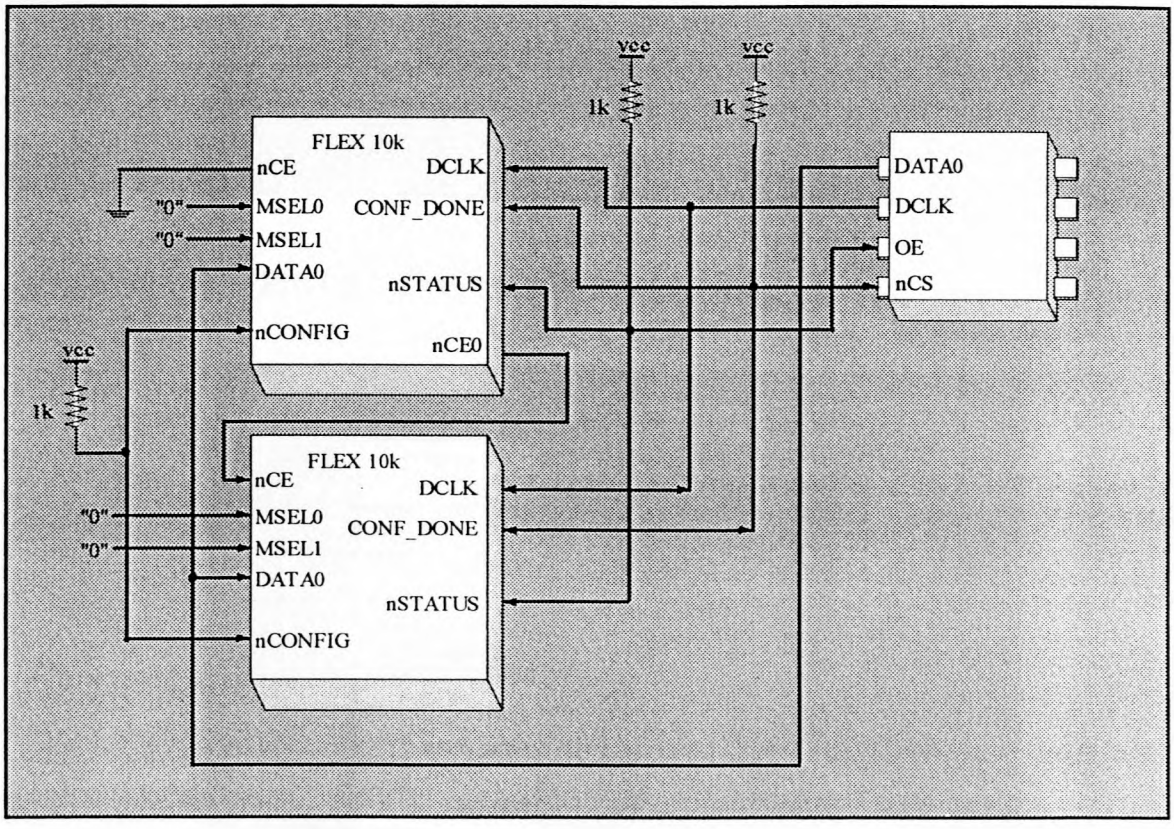


FIGURE 2.20 : PIN CONNECTIONS FOR CONFIGURATION OF FLEX 10K DEVICE/S USING AN EPROM

During the development stage of this project, the byte blaster is used to upload the data into the FLEX 10K devices. The configuration scheme is again the passive serial, where the byte blaster provides the clock signal for serial data transfer. Figure 2.21 indicates the pin connections used for PS byte blaster configuration of both single and multiple FLEX 10K devices.

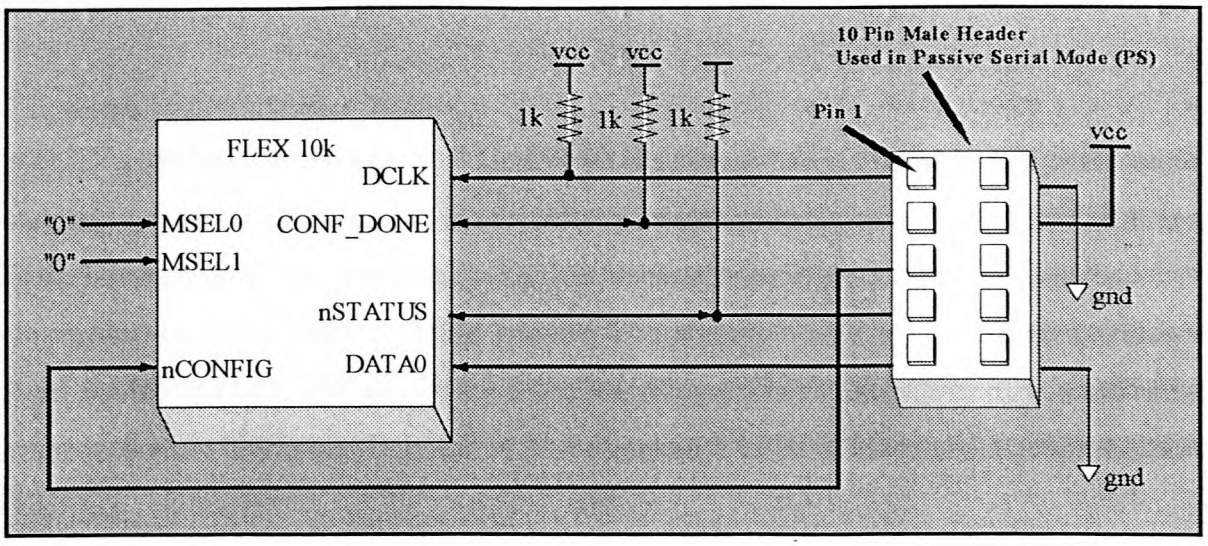


FIGURE 2.21a : SINGLE FLEX 10K DEVICE

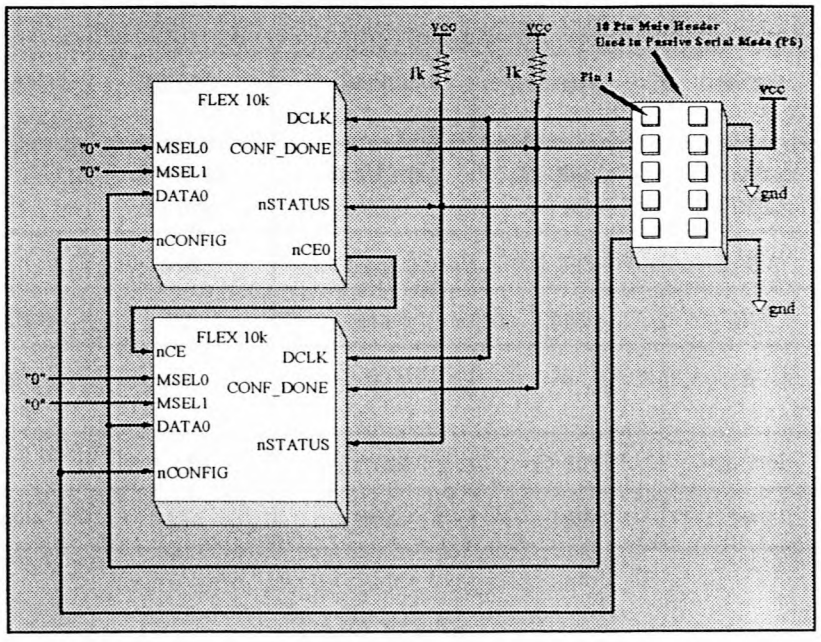


FIGURE 2.21b : MULTIPLE FLEX 10K DEVICES

Altera's MAX+PLUS II software is used to generate the necessary files for configuring the FLEX 10K devices. If the byte blaster is used for configuration, an SRAM object file (.sof) is generated by MAX+PLUS II. If the serial EPROM is used for configuration, a programmer object file (.pof) is generated by MAX+PLUS II.

2.1.10.8 *FLEX 10K Architecture*

– *Logic Implementation Structure*

The FLEX 10K family is the first PLD family that is designed to implement designs that integrate a complete system on a device. [33] Altera offers a range of speed grades in the FLEX 10K range with the latest speed grades allowing FLEX 10K devices to achieve the best high-density performance in the PLD market. Table 2.5a and 2.5b presents the FLEX 10K family features. The SRAM based FLEX 10K device has a flexible, programmable embedded array which consists of embedded array blocks (EABs). These EABs contain 2-k bits of RAM (4-k bits for the FLEX 10 KE).

FLEX 10K Device Features					
Feature	EPF10K10 EPF10K10A	EPF10K20	EPF10K30 EPF10K30A	EPF10K40	EPF10K50 EPF10K50V
Typical gates (logic and RAM), Note (1)	10 000	20 000	30 000	40 000	50 000
Usable gates	7 000 to 31 000	15 000 to 63 000	22 000 to 69 000	29 000 to 93 000	36 000 to 116 000
Logic elements (LEs)	576	1 152	1 728	2 304	2 880
Logic array blocks (LABs)	72	144	216	288	360
Embedded array blocks (EABs)	3	6	6	8	10
Total RAM bits	6 144	12 288	12 288	16 384	20 480
Maximum user I/O pins	134	189	246	189	310

TABLE 2.5a

FLEX 10K Device Features				
Feature	EPF10K70	EPF10K100 EPF10K100A	EPF10K130V	EPF10K250A
Typical gates (logic and RAM) Note (1)	70 000	100 000	130 000	250 000
Usable gates	46 000 to 118 000	62 000 to 158 000	82 000 to 211 000	149 000 to 310 000
LEs	3 744	4 992	6 656	12 160
LABs	468	624	832	1 520
EABs	9	12	16	20
Total RAM bits	18 432	24 576	32 768	40 960
Maximum user I/O pins	358	406	470	470

Note of tables
(1) For designs that require JTAG boundary-scan testing, the built-in JTAG circuitry contributes up to 31 250 additional gates.

TABLE 2.5b

A block diagram illustrating the internal structure of the FLEX 10K device is presented in Figure 2.22.

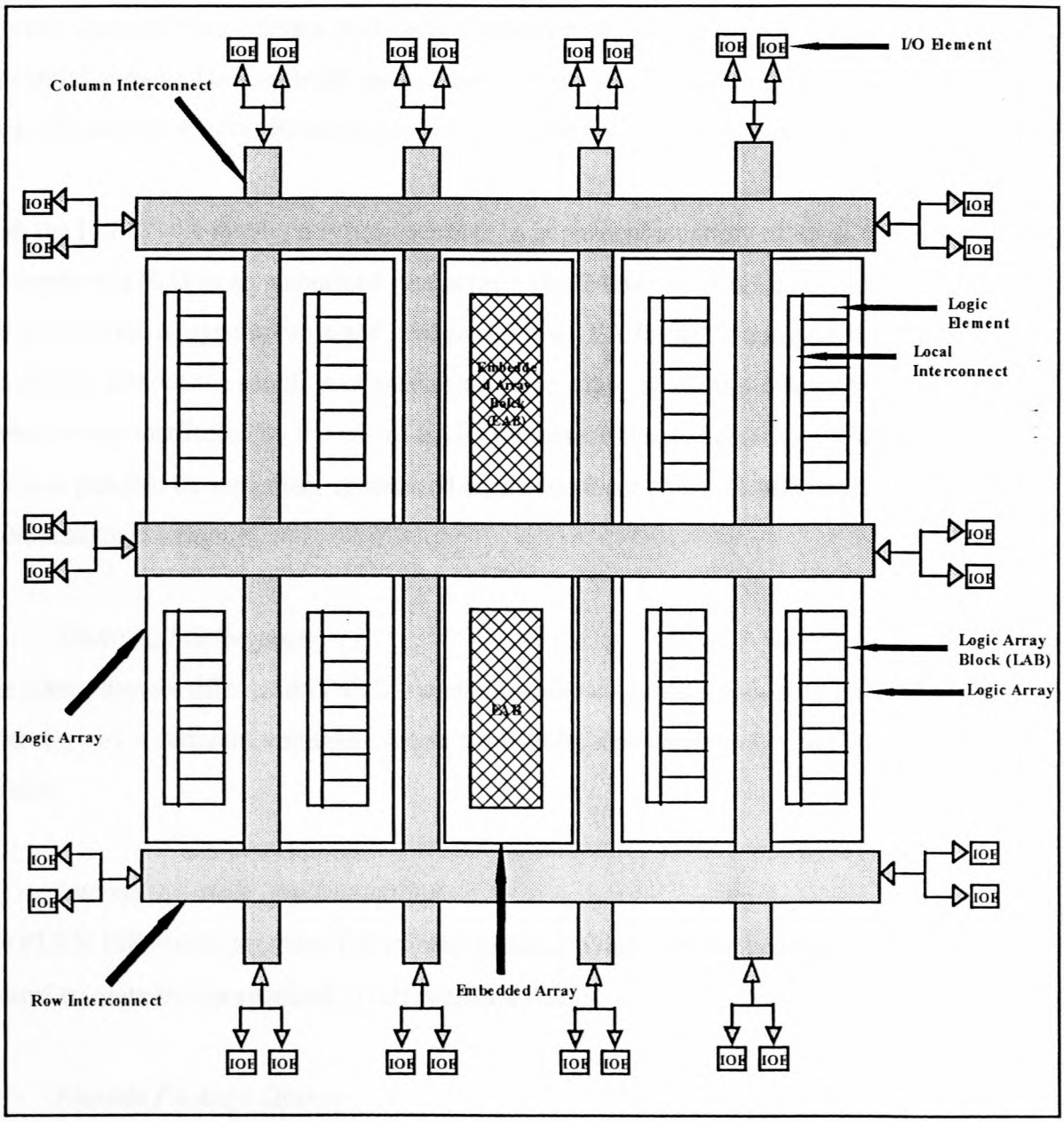


FIGURE 2.22 : DIAGRAM OF INTERNAL STRUCTURE OF FLEX 10K DEVICE

The logic array consists of logic array blocks (LABs) which communicate through a fully populated local interconnect structure. The LAB consists of Logic Elements (LEs). Each LE contains a four-input look-up table (LUT) and a programmable register with a local clock enable control. Each LE then offers a dual-output structure that allows for both combinational and sequential output. The fast track interconnect routing structure provides communication for both the LABs and the EABs. Each interconnect row and column feeds multiple I/O elements (IOEs).

With the FLEX 10K family, it is now possible to implement a variety of mega-functions as efficiently in a PLD as an embedded gate array. The EABs can be used to create ROMs, FIFOs and asynchronous, synchronous and dual-port RAM. EABs can be configured for various width and depths without the sacrifice of speed. All logic array capabilities offered by teh FLEX 10K device remain unaffected by the use of EABs for memory implementation. With the use of EABs, it is now possible to implement specialized arithmetic functions more efficiently than with traditional logic arrays.

– *Multivolt I/O Support*

The Altera devices offer MultiVolt I/O operation, allowing these devices to interface directly with other devices of different voltages. Altera's 5,0 Volt range can interface with 3,3 V and 5,0 V devices.

– *Internal Tri-state Implementation*

The FLEX 10K family provides full tri-state emulation that enables the implementation of the internal tri-state busses required in this project's design.

– *Flexible Package Option*

Altera offers a range of FLEX 10K devices which support pin-compatibility within the same packages. This project requires the use of the FLEX 10K10 device in the QFP package. The pin-compatibility support offered by Altera means that it allows for migration from a 10 000 gate device (FLEX 10K10) to the FLEX 10K20, 10K30, 10K40 and 10K50 devices in the same QFP package.

This provides the designer with the option of modifying the PLDs internal design to such an extent that the densities may increase by 400% while no modifications are required to the PCB.

2.1.10.9 *Selection Criteria*

A number of factors contributed to the selection of the FLEX 10K device for this project. The logic density of the FLEX 10K10 with its 10 000 gates was the smallest component which could contain the design. The PCB designed for this project has support for FLEX 10K devices. They are configured in cascade, with the 16-bit address/data bus passing through both devices. The design methodology, along with the migration support offered for the FLEX 10K devices, allows the logic density to vary from 2x 10 000 gates to 2x 50 000 gates, using the QFP 144 pin package.

The FLEX 10K family was primarily chosen for its support of mega-function implementation. This technology allows for the incorporation of many off-the-shelf building blocks that implement useful functions such as processors, DSPs, bus controllers and interfaces.

The bus must therefore be able to present a tri-stated state and this requires the feature offered by the FLEX 10K device.

2.2 **The Support Software**

All of the source code for this thesis is contained in the CD appended to this thesis.

2.2.1 ***80C196X A96 Compiler***

An overview of the application development process offered by the BSO Tasking software is presented in figure 2.23.

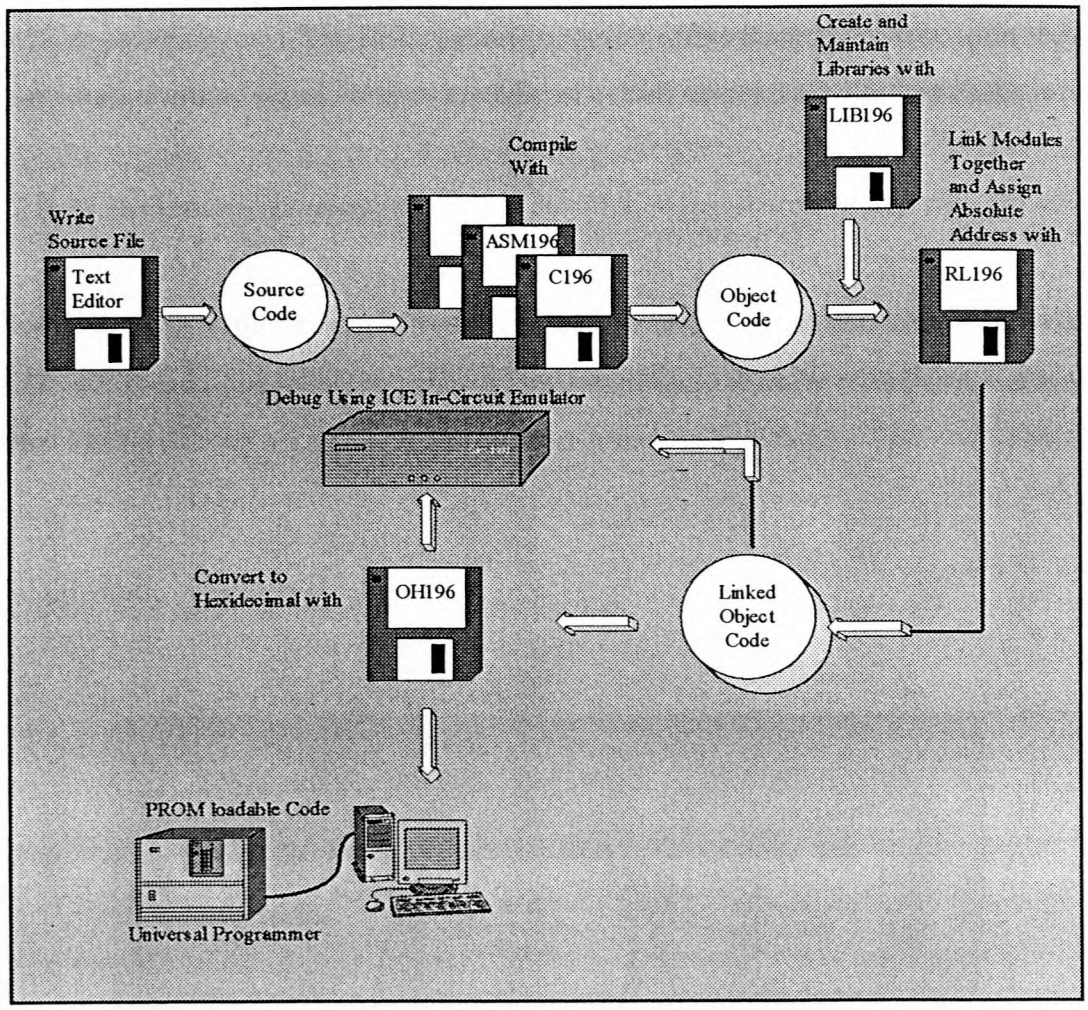


FIGURE 2.23 : BSO TASKING'S APPLICATION DEVELOPMENT PROCESS

The ASM 196 compiler of BSO Tasking is used to compile the files for microcontroller initialization (startup.A96), temporary register initialization (tmpreg.A96) and LCD initialization (LCDinit.A96). The reason for the use of assembly language rather than the C language for implementation code, is for the purpose of speed optimization.

2.2.2 *80C196X C196 Compiler*

The Ansi-C compiler of BSO Tasking is used to compile the source code for the 196 microcontroller-based system. The BSO Tasking compiler offers floating point support for the 80C196. A comprehensive set of libraries are offered as part of the BSO Tasking package.

2.2.3 *PLD Programming Support*

Cypress's WARP PLD programming software is used to compile, synthesize and simulate the VHDL code for the GALS used in the PHASE 1 design. Figure 2.24 illustrates the development process used by WARP.

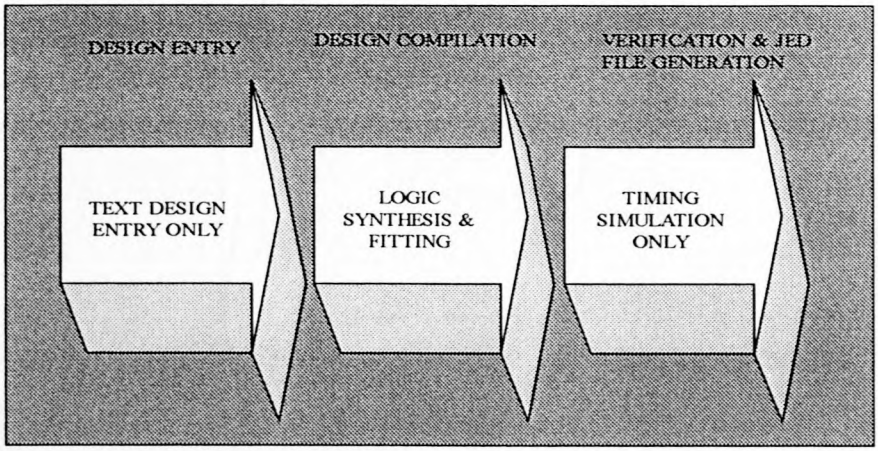


FIGURE 2.24

2.2.4 *Altera's MAX+PLUS II*

The MAX+PLUS II programmable logic development system is used in the design of the PHASE 2 CPLD supported system. This provides an architecture independent design environment which ensures easy design entry, fast compilation and uncomplicated device programming [7]. Figure 2.25 illustrates the MAX+PLUS II design environment.

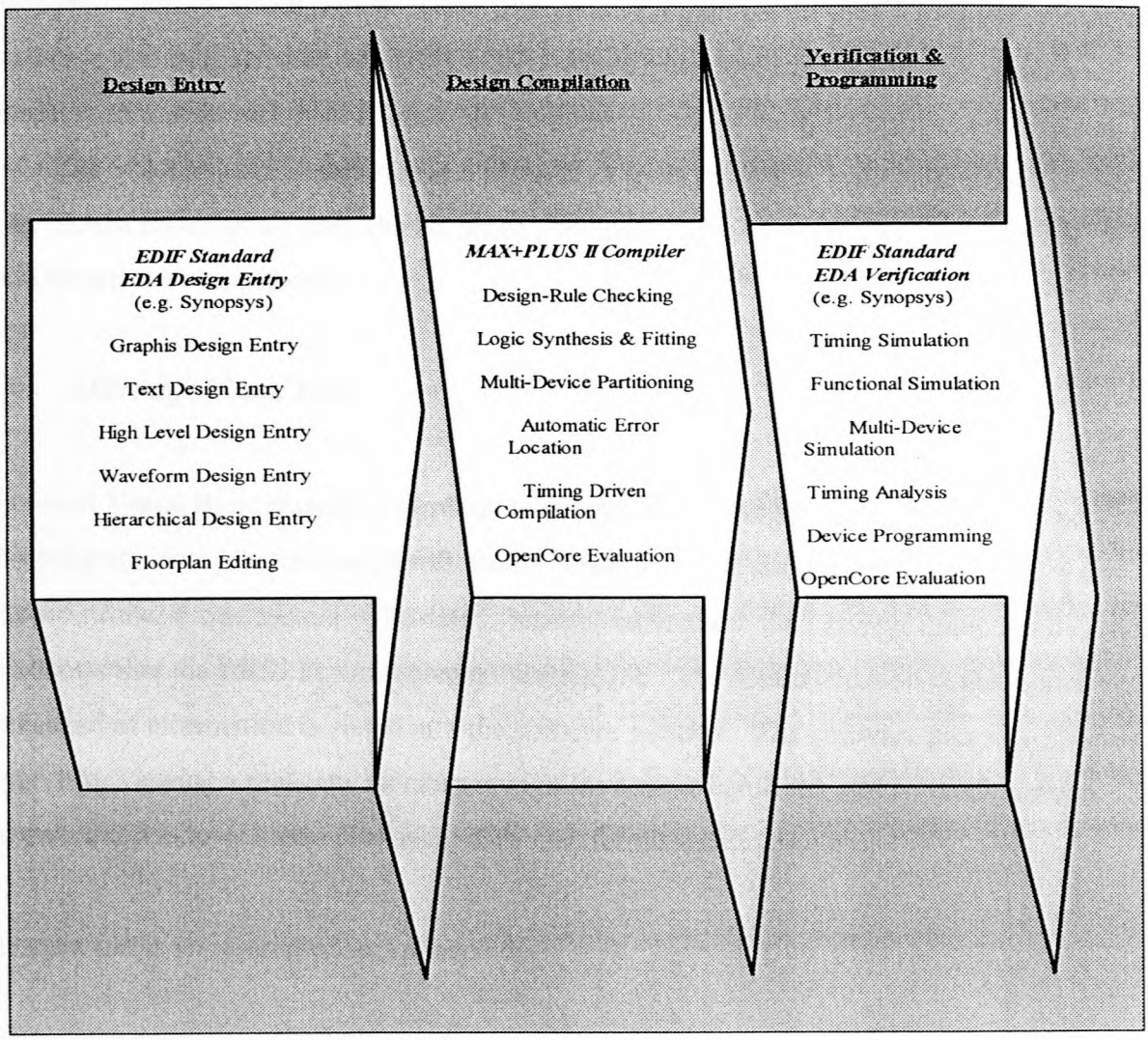


FIGURE 2.25 : THE MAXPLUS II DEVELOPMENT CYCLE

MAX+PLUS II allows designers to combine text, graphic, and waveform design entry methods while creating hierarchial single- or multi-device designs. The MAX+PLUS II compiler performs minimization and logic synthesis, then lists the design into either a single or multiple devices, and finally generates programming or configuration data.

The software design tool also offers design verification with functional and timing simulation with delay prediction for speed critical paths. Simulation for multiple devices across multiple device families is also supported by MAX+PLUS II. Altera's MAX+PLUS II software provides interfaces to a wide variety of EDA tools. Synopsys is one of many companies with such an EDA tool. The MAX+PLUS II software and these EDA tools share information via their EDIF netlist files, SRAM object files (.sof), the library of parameterized modules (LPM), verilog HDL, VHDL and Designware components.

2.2.5 *Microsoft Visual Basic*

Microsoft Visual Basic is used to develop a windows based fuel management software program. This program incorporates a microsoft access database, which stores data obtained from the microcontroller-based system. The data is transferred between the personal computer and the microcontroller via a RS232 serial communication link. The communication controls and data packatized of information is coded into the software package. The software program also has support for viewing a real-time memory map of the remote controller and its configuration. The program has windows forms displaying data used for diagnostic purposes.

The main forms are presented in Appendix K.

Third-party Custom controls used in the development of the program include:

- PDQ Comm controls
- Dallas Tag Communication Controls
- Real-Time Graphics Controls
- Copy-Protection Controls

2.2.6 *Microsoft Visual C++*

Due to the fact that Visual Basic did not support the binary manipulation required to interpret the packatized data obtained from the microcontroller, additional math functions had to be created in Dynamic Linked Libraries (DLLs). These DLLs are developed in the Microsoft Visual C++ development environment.

2.2.7 *Tango Design Package*

The schematic PHASE 1 of the microcontroller electronic design and all of it's peripherals were created using the Tango Schematic Entry software. The PHASE 1 printed circuit boards were created using the Tango PCB software. The routing of the PHASE 1 PCBs was completed using the Tango Router.

2.2.8 *Protel Design Package*

The schematic, PCB and routing of the PHASE 2 design is completed using the Protel 99 development software.

Chapter 3

Design of a PHASE 1 Microcontroller-based System using Gate-level Logic Components

This chapter will review the design of the system consisting of the components illustrated in Figure 3.1.

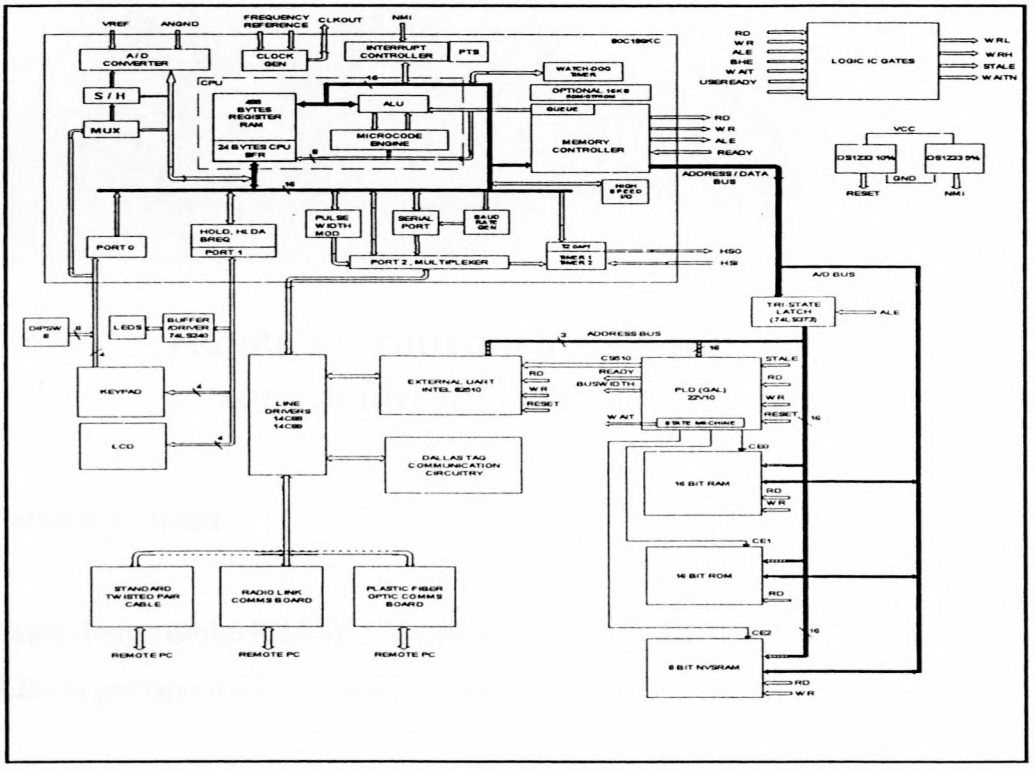


FIGURE 3.1 PHASE 1 MICROCONTROLLER BASED SYSTEM

Figure 3.2 presents the completed design of the PHASE 1 microcontroller-based system.

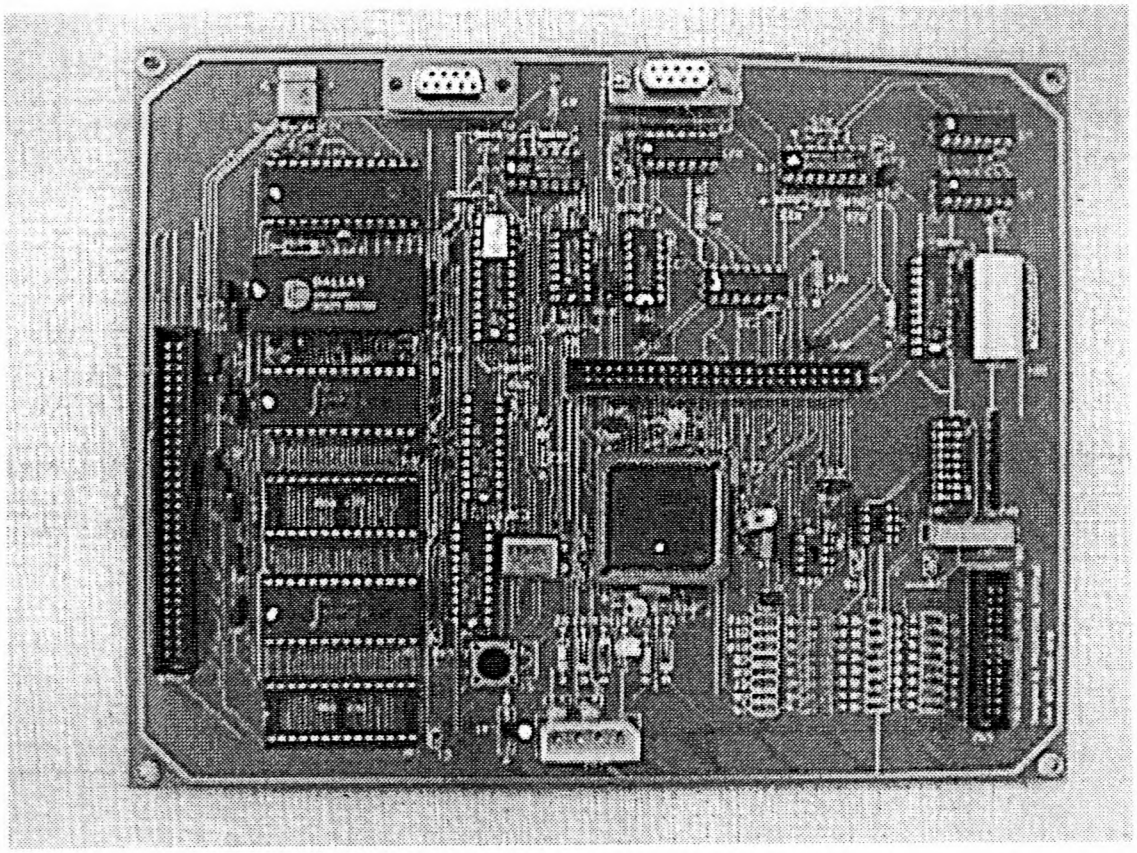


FIGURE 3.2 : PHOTOGRAPH OF THE PHASE 1 MICROCONTROLLER-BASED SYSTEM

3.1 Hardware Design

The hardware design of the PHASE 1 system makes use of gate-level logic components with simple PLDs to perform the function of system control.

3.1.1 *Intel 80C196KC Microcontroller*

The 80C196KC, 16-bit microcontroller is used in this design because of its comprehensive peripheral support. The internal peripheral components are presented as implemented in this design.

3.1.1.1 *Design Considerations*

– *Chip Configuration Byte (CCB)*

This is the first byte fetched from memory after a device reset. A reset loads the CCB into the Chip Configuration Register (CCR). The CCR is presented in Figure 3.3.

7	6	5	4	3	2	1	0
1	1	1	1	1	1	1	1

Bit	Function	
0	= PD	(Power-down Enable)
1	= BW0	(Buswidth Control)
2	= WR	(Write Select Strobe Mode)
3	= ALE	(Select Address Valid Strobe Mode)
4	= IRC0	(Internal Ready Control)
5	= IRC1	Internal Ready Control)
6	= LOC0	(Lock Bit 0)
7	= LOC1	(Lock Bit 1)

FIGURE 3.3 : CHIP CONFIGURATION REGISTER

The CCR control the internal READY mode, bus control signals, bus width operations and power-down modes. The CCB is located at address 2018H. A value of FFH is written into the CCB on power-up. This byte configures the microcontroller to allow for wait-states controlled by the external READY pin.

– *Horizontal and Vertical Windowing*

The 8XC196 supports both horizontal and vertical windowing which implements a scheme of swapping three 24 byte memory blocks within the lowest 24 bytes of the lower register file. The horizontal windows with their addresses are presented in Figure 3.4.

	HWINDOW 0 (Read)	HWINDOW 0 (Write)	HWINDOW 1(Read/Write)
17H	IOS2	PWMO_CONTROL	PWM2_CONTROL
16H	IOS1	IOC1	PWM1_CONTROL
15H	IOS0	IOC0	Reserved
14H	WSR	WSR	WSR
13H	INT_MASK1	INT_MASK1	INT_MASK1
12H	INT_PEND1	INT_PEND1	INT_PEND1
11H	SP_STAT	SP_CON	Reserved
10H	IOPORT2	IOPORT2	Reserved
0FH	IOPORT1	IOPORT1	Reserved
0EH	IOPORT0	BAUD_RATE	Reserved
0DH	TIMER2 (HI)	TIMER2 (HI)	Reserved
0CH	TIMER2 (LO)	TIMER2 (LO)	IOC3
0BH	TIMER1 (HI)	IOC2	Reserved
0AH	TIMER 1 (LO)	WATCHDOG	Reserved
09H	INT_PEND	INT_PEND	INT_PEND
08H	INT_MASK	INT_MASK	INT_MASK
07H	SBUF (RX)	SBUF (TX)	PTSSRV (HI)
06H	HSI_STATUS	HSO_COMMAND	PTSSRV (LO)
05H	HSI_TIME (HI)	HSO_TIME (HI)	PTSSEL (HI)
04H	HSI_TIME (LO)	HSO_TIME (LO)	PTSSEL (LO)
03H	AD_RESULT (HI)	HSI_MODE	AD_TIME
02H	AD_RESULT (LO)	AD_COMMAND	Reserved
01H	ZERO_REG (HI)	ZERO_REG (HI)	ZERO_REG (HI)
00H	ZERO_REG (LO)	ZERO_REG (LO)	ZERO_REG (LO)

	HWINDOW 15 (Read)	HWINDOW 15 (Write)
17H	PWM0_CONTROL	IOS2
16H	IOC1	IOS1
15H	IOC0	IOS0
14H	WSR	WSR
13H	INT_MASK1	INT_MASK1
12H	INT_PEND1	INT_PEND1
11H	SP_CON	SP_STAT
10H	Reserved	Reserved
0FH	Reserved	Reserved
0EH	Reserved	Reserved
0DH	T2CAPTURE (HI)	T2CAPTURE (HI)
0CH	T2CAPTURE (LO)	T2CAPTURE (LO)
0BH	IOC2	TIMER1 (HI)
0AH	WATCHDOG	TIMER1 (LO)
09H	INT_PEND	INT_PEND
08H	INT_MASK	INT_MASK
07H	SBUF (TX)	SBUF (RX)
06H	HSO_COMMAND	HSI_STATUS
05H	HSO_TIME (HI)	HIS_TIME (HI)
04H	HSO_TIME (LO)	HSI_TIME (LO)
03H	HSI_MODE	AD_RESULT (HI)
02H	AD_COMMAND	AD_RESULT (LO)
01H	ZERO_REG (HI)	ZERO_REG (HI)
00H	ZERO_REG (LO)	ZERO_REG (LO)

FIGURE 3.4 :80C196KCs HORIZONTAL WINDOWS

Vertical windowing is not used in this design.

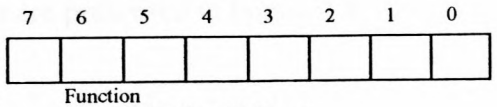
3.1.1.2 *Standard I/O Ports*

The 80C196KC has five 8-bit I/O ports. The port pin functions are listed in Table 3.1

Port Pin(s)	Port Type	Alternate Function	Alternate Type	Control SFR
P0	Input	ACH0-ACH7	Input	
P1.0, P1.1, P1.2	Quasi-Bidirectional	None		
P1.3, P1.4	Quasi-Bidirectional	PWM1, PWM2	Output	IOC3
P1.5, P1.6	Quasi-Bidirectional	BREQ#, HLDA#	Output	WSR
P1.7	Quasi-Bidirectional	HOLD	Input	WSR
P2.0	Output	TXD	Output	IOC1
P2.1	Input	RXD	Input or Output	SPCON
P2.2	Input	EXTINT	Input	IOC1
P2.3	Input	T2CLK	Input	IOC0
P2.4	Input	T2RST	Input	IOC0
P2.5	Output	PWM0	Output	IOC1
P2.6	Quasi-Bidirectional	T2UP-DN	Input	IOC2
P2.7	Quasi-Bidirectional	T2CAPTURE	Quasi-Bidirectional	
P3, P4	Open-Drain Bidirectional	AD0-AD15	Bidirectional	

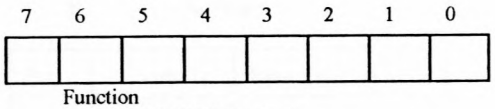
TABLE 3.1 PORT PIN FUNCTIONS

The configuring of the relevant port pins is done by writing to the input/output control registers (IOCs). Figure 3.5 presents the four control registers IOC0, IOC1, IOC2 and IOC3 with their configurations as used in this design.



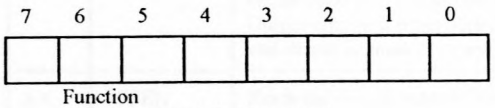
IOC0

Bit	Function
0	= Enable HSI.0 as HSI input
1	= Timer 2 software reset
2	= Enable HSI.0 as HSI input
3	= Timer 2 external reset source
4	= Enable HSI.0 as HSI input
5	= Timer 2 reset source
6	= Enable HSI.0 as HSI input
7	= Timer 2 clock source



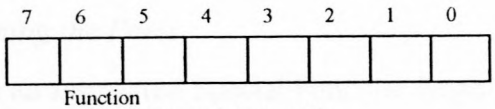
IOC1

Bit	Function
0	= Select P2.5/PWM output
1	= Select external interrupt INT07 source
2	= Enable timer 1 overflow interrupt
3	= Enable timer 2 overflow interrupt
4	= Enable HSO.4 pin as output
5	= Select P2.0/TXD output
6	= Enable HSO.05 pin as output
7	= Select HSI interrupt source



IOC2

Bit	Function
0	= Enable timer 2 fast increment
1	= Enable timer 2 up/down count
2	= Enable PWM clock prescaler
3	= Enable AD_TIME register
4	= Disable A/D clock prescaler
5	= Select timer 2 overflow boundary
6	= Enable locked CAM entries
7	= Clear all CAM entries

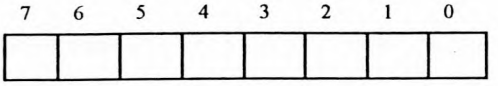


IOC3

Bit	Function
0	= Timer 2 internal clock enable
1	= Reserved / CLKOUT disable
2	= PWM1 select
3	= PWM2 select
4-7	= ----

FIGURE 3.5 : I/O CONTROL REGISTERS

The port registers are presented in Figure 3.6.



0 = Port Pin 0
1 = Port Pin 1
2 = Port Pin 2
3 = Port Pin 3
4 = Port Pin 4
5 = Port Pin 5
6 = Port Pin 6
7 = Port Pin 7

Port	Accessed by	Read operation	Write operation
0	IOPORT0	Reads the current value of the port pins. The port should not be read while an A/D conversion is in progress.	Not possible. Writing to IOPORT0 does not affect Port 0, but does affect the baud rate controller.
1	IOPORT1	Reads the current value of the port pins. A port pin may have been written a value of one but return a value of zero because it is being overridden externally.	Writes change the state of the port pin latch. Changing the port pin latch has no effect on a port pin if its alternate function has been selected.
2	IOPORT2	Reads the current value of the port pins, except those pins that are output only. Output-only pins return an unspecified value and should be masked by software.	Writes change the state of the port pin latch, except those pins that are input only. Input-only port pins have no pin latches, so the value written is ignored.
3,4	1FFEH	Reads the current value of the port pins. A port pin may have been written a value of one but return a value of zero because it is being overridden externally. Ports 3 and 4 are always accessed together as a word read operation.	Changes the state of the port pin latch. Any external access of the bus controller overrides the value of the port pin latch to perform the bus operation. Ports 3 and 4 are always accessed together as a word write operation.

FIGURE 3.6 : 80C196KC PORT REGISTERS

– *Programming the Ports*

Each I/O port has an associated Special Function Register (SFR) which is used to read or write the port pins. Not all the port SFRs are readable and writeable. Port 0 is only readable. Port 1 is readable and writeable. Port 2 is readable and writeable. The quasi-bidirectional ports 3 and 4 are also readable and writeable. When the value of an output pin is read, it's value is undefined and should be masked. The port addresses are as follows:

I/O port0 → 0x000EH
I/O port1 → 0x000FH
I/O port2 → 0x0010H

– *Port 0*

Port 0 is an input port that is also an analog input for the A/D converter. Bits 0, 1, 2 and 3 are used to read in the columns off the keypad, while bits 4, 5, 6 and 7 are used to read in the status of the pump nossels (nossel switches). All of port 0 is also connected in parallel to an 8-bit bank of DIP switches. These switches are used to establish board diagnostic routines when the microcontroller powers-up. The keyboard connections are illustrated in Figure 3.7.

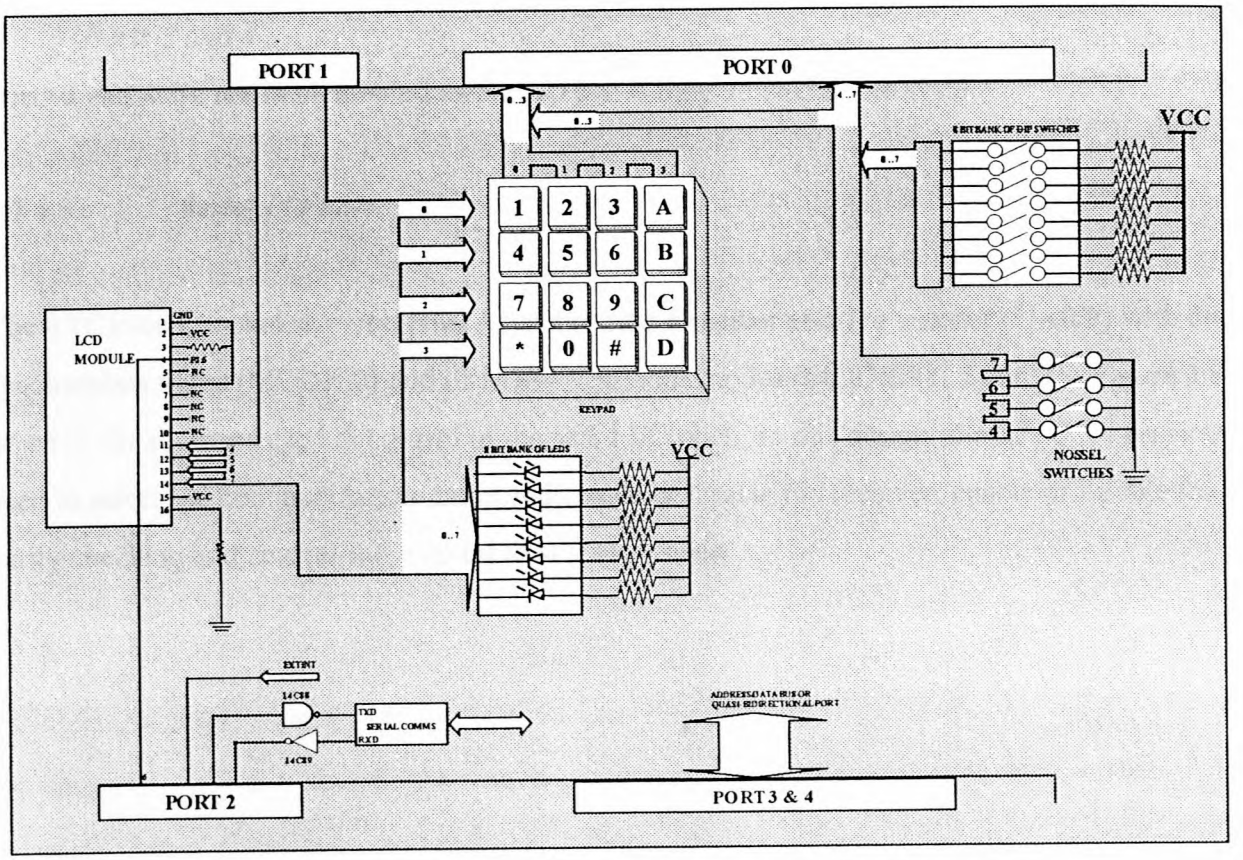


FIGURE 3.7 : PHASE 1 DESIGN PORT CONNECTIONS

– *Port 1*

Port 1 is an input/output with bits 0, 1, 2 and 3 writing out to the 4 rows of the keypad. Bits 4, 5, 6 and 7 are used to write data to the LCD which is implemented in 4-bit mode. The LCD also has full support for 8-bit data transfer but due to the port constraints, it was decided to use the 4-bit mode. A bank of 8-bit leds is connected in parallel to port 1 as illustrated in Figure 3.7.

– *Port 2*

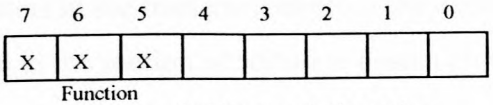
Port 2 is an input/output port with many alternate functions as presented in Table 3.1. The timers (timer 1 and 2) are used to generate interrupts for timed reading of ports as system status monitors. Bit 6 is used as a control bit for the LCD, while bit 2 is used to hold the external interrupt status. Bits 0 and 1 are the transmit and receive status bits used for the internal serial communication port.

– *Ports 3 and 4*

Ports 3 and 4 are not implemented as the AD bus is fully utilized in the system.

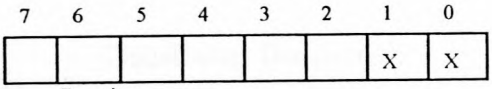
3.1.1.3 *Serial I/O Port*

The 8XC196KC includes a Universal Asynchronous Receiver and Transmitter (UART) with one synchronous mode (MODE 0) and three asynchronous modes (MODES 1, 2 and 3). Figure 3.8 presents the registers used in control of the serial I/O port. In this design, the SP_CON register is used to select the communication mode and enable or disable the receiver, enable or disable the parity checking and control the nine-bit data transmission.



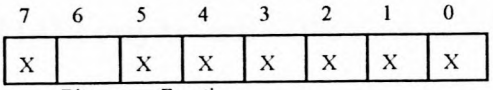
SP_CON Register

Bit	Function
0, 1	= Serial communication mode
2	= Even parity enable
3	= Receive enable
4	= 7B8 (transmit bit 8) flag
5, 6, 7	= Reserved bits



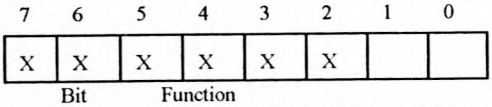
SP_STAT Register

Bit	Function
0, 1	= Reserved bits
2	= Overrun error bit (OE)
3	= Transmitted empty (TXE)
4	= Framming error (FE)
5	= Transmit interrupt (TI)
6	= Receive interrupt (RI)
7	= Receive parity error (P1PE) / Receive bit 8 (RB8)



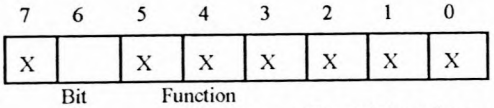
INT_MASK

Bit	Function
6	= Enable or masks the serial port interrupt



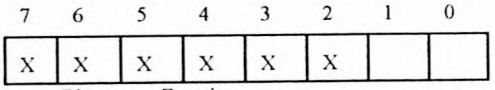
INT_MASK1

Bit	Function
0	= Enables or masks the transmit interrupt
1	= Enables or mask the receive interrupt



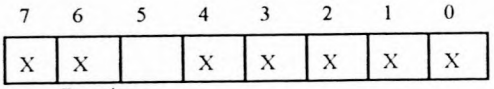
INT_PEND

Bit	Function
6	= Indicates a pending serial port interrupt and is cleared on interrupt service



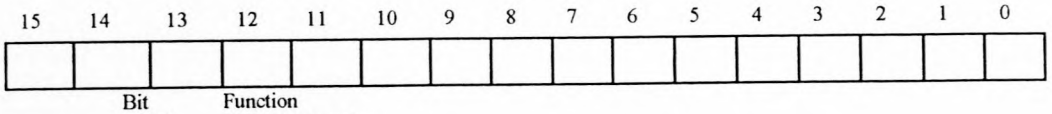
INT_PEND1

Bit	Function
0	= Indicates pending transmit interrupt and is cleared on interrupt service
1	= Indicates pending receive interrupt and is cleared on interrupt service



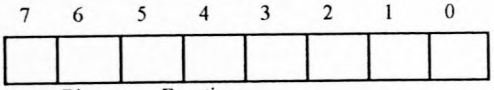
I/OC1

Bit	Function
5	= Enable/disable the TXD function of port 2 bit 0



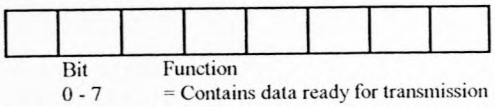
BAUD_RATE

Bit	Function
0 - 14	= Baud rate value
15	= Selects the clock source



SBUF (RX)

Bit	Function
0 - 7	= Contains the data received from the serial port



SBUF (RX)

Bit	Function
0 - 7	= Contains data ready for transmission

FIGURE 3.8 : CONTROL REGISTERS FOR SERIAL COMMUNICATION PORT

Use of these registers in the implementation of the internal serial port can be seen in the flow diagram presented in the section of software design on the PHASE 1 system. The baud rate of the communication port can be set by entering a value into the baud rate register. The required value used in this design was calculated as follows:

$$\text{BAUD VALUE} = \frac{\text{Oscillator frequency}}{\text{Baud rate x 16}} - 1 = \frac{20\ \text{MHz}}{9600\text{x}16} - 1$$
$$= 129$$

Hence the value entered into the Baud_Rate Register is 8081 Hex, since the most significant bit is set in order that XTAL1 may be used as the clock source for the baud_rate generator.

3.1.1.4 *Interrupts*

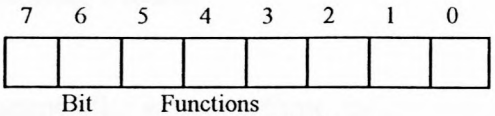
Table 3.2 presents all the interrupts used in this design.

Number	Interrupt Vector	Source(s)	Interrupt Vector Location	PTS Vector Location	Priority (1)
Special	Unimplemented Opcode	Unimplemented Opcode	2012H	—	—
Special	Software Trap	TRAP Instruction	2010H	—	—
INT15	NMI (2)	NMI	203EH	—	15
Each of the following, maskable interrupts can be assigned to the PTS Any PTS interrupt has priority over all other maskable interrupts					
INT14	HSI FIFO Full	HSI FIFO Full	203CH	205CH	14
INT13	EXTINT1 (2)	P2.2	203AH	250AH	13
INT12	Timer 2 Overflow	Timer 2 Overflow	2038H	2058H	12
INT11	Timer 2 Capture (2)	Timer 2 Capture	2036H	2056H	11
INT10	HSI FIFO 4	HSI FIFO Fourth Entry	2034H	2054H	10
INT09	Receive	RI Flag (3)	2032H	2052H	9
INT08	Transmit	TI Flag (3)	2030H	2050H	8
INT07	EXTINT (2)	P2.2 or P0.7	200EH	204EH	7
INT06	Serial Port	RI Flag and TI Flag (4)	200CH	204CH	6
INT05	Software Timer	Software Timer 0-3 Timer 2 Reset A/D Conversion Start	200AH	204AH	5
INT04	HSI.0 Pin (2)	HSI.0	2008H	2048H	4
INT03	High Speed Outputs	HSO.0-HSO.5	2006H	2046H	3
INT02	HSI Data Available	HSI FIFO Full or HSI Holding Reg. Loaded	2004H	2044H	2
INT01	A/D Conversion Complete	A/D Conversion Complete	2002H	2042H	1
INT00	Timer Overflow	Timer 1 or Timer 2	2000H	2040H	0

TABLE 3.2 : INTERRUPT VECTOR SOURCES, LOCATIONS AND PRIORITIES

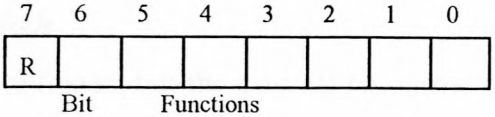
When the microcontroller detects an interrupt, it generates and executes a special interrupt call. The contents of the program counter are pushed onto the stack after which the contents of the appropriate interrupt vector are loaded. The upper and lower interrupt vectors in the special purpose memory contain the addresses of the interrupt service routine. The CPU executes the interrupt service routine at the relevant address. Once the service routine has been completed, the program counter is reloaded from the stack allowing program execution to continue.

The priorities of the interrupts are as indicated in Table 3.2. The interrupt priorities are altered in this design by controlling the interrupt mask registers. The registers flagging and controlling the interrupts are presented in Figure 3.9.



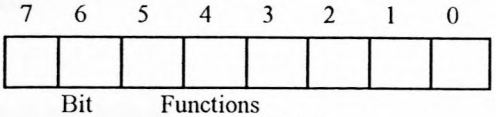
INT_MASK

Bit	Functions
0	= TIMER_MASK
1	= AD_MASK
2	= HSIDAT_MASK
3	= HSO_MASK
4	= HSI0_MASK
5	= SWT_MASK
6	= SER_MASK
7	= EXTINT_MASK



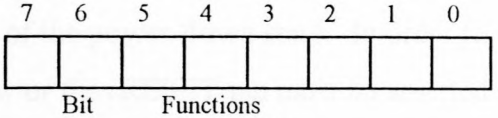
INT_MASK1

Bit	Functions
0	= TI_MASK
1	= RI_MASK
2	= HS14_MASK
3	= T2CAP_MASK
4	= T2OVF_MASK
5	= EXTINT1_MASK
6	= FIFO_MASK
7	= NMI_MASK



INT_PEND

Bit	Functions
0	= TIMER_PEND
1	= AD_PEND
2	= HSIDAT_PEND
3	= HSO_PEND
4	= HSI0_PEND
5	= SWT_PEND
6	= SER_PEND
7	= EXTINT_PEND



INT_PEND1

Bit	Functions
0	= TI_PEND
1	= RI_PEND
2	= HS14_PEND
3	= T2CAP_PEND
4	= T2OVF_PEND
5	= EXTINT1_PEND
6	= FIFO_PEND
7	= NMI_PEND

FIGURE 3.9 : INTERRUPT CONTROL REGISTERS

The interrupt pending registers set their bits as flags to indicate the occurrence of the relevant interrupts.

the 8XC196 microcontroller supports three special operating modes. The Idle, Power-down and On-circuit Emulation (ONCE). All three modes are included in this design. Both the Idle and Power-down modes reduce the power consumption while the ONCE mode is a test mode that electrically isolates the chip from other system components.

– *The Idle Mode*

The CPU clocks are frozen, but the external peripheral clocks (CLKOUT) remain active, allowing the external devices to maintain operation if required. No instruction are executed, but internal RAM, timers, counters, serial ports and interrupt systems remain functional. This reduces the microcontroller's power consumption by about 40%. During this mode, the address/data bus floats externally. The IDLPD #1 (idle/power-down) software instruction controls the entry or exit of the idle state.

– *The Power-down Mode*

The power-down mode places the microcontroller into a very low power state. In this mode, all of the peripherals are powered down and the external peripheral clock (CLKOUT) is frozen at logic state zero. The power consumption is reduced to the microwatt range. During this mode, the address/data bus floats externally. The IDLPD #2 (idle/power-down) software instruction controls the entry of the power-down state. In order to exit the power-down state, the V_{PP} pin must be driven low or the /RESET pin must be asserted or the external interrupt must be generated. The power-down mode is disabled when the PD bit (bit 0) of the CCR is cleared (refer to Figure 3.2).

– *The ONCE Mode*

This mode is a test mode used to electrically isolate the microcontroller from other devices on the PCB. This mode is used purely for test purposes in the PHASE 1 design, but the PHASE 2 design requires this mode while the CPLDs are being configured. All of the pins, except the XTAL1 and XTAL2, have weak pull-ups or pull-downs and are not high impedance. The /RESET pin must remain high during the ONCE mode execution. The ONCE mode is entered if port 2 bit 0 is held low during the rising edge of /RESET. To exit the ONCE mode, the port 2 bit 0 must float while the /RESET pin is asserted.

3.1.1.6 *Timing Considerations and Calculations*

The timing diagrams and analysis of the PHASE 1 design is presented in Appendix A. All the calculated values are based on the 80C196KC operating at 20 MHz. The values are taken from the November 1994 version of the 80C196KC Commercial/Express CHMOS microcontroller data sheet. The Intel order number is 270942-005.

3.1.2 ***Programmable Logic Devices (PLDs)***

The PLDs used in the PHASE 1 design are the 22V10Ds from Cyprus. This Generic Array Logic Device (GAL) is chosen for its pin support and its internal capacity. The GAL is used to provide the additional control signals required for the system to operate. The GAL also allows for the implementation of a state machine for generating the required wait-states. These wait-states are used in conjunction with external logic to lengthen the ALE pulses as well. This provides for full extended access times supporting a range of components.

3.1.2.1 *Control Signals*

The capacity requires that two GALs be used to implement the required logic support. Tables 3.3 and 3.4 present the control signals as used in both GALs. Appendix C presents a detailed memory map of the peripheral control signals. The physical connections of both GALs are illustrated in the schematic of the PHASE 1 design presented in Appendix L.

Signal	Type		Function	Address
CLKOUT	INPUT	CLOCK	8XC196KC system clockout	---
STALE	INPUT	ACTIVE HIGH	Stretched address latch enable	---
/HOLDA	INPUT	ACTIVE LOW	8XC196KC hold acknowledge	---
A8	INPUT	ACTIVE HIGH	Latched address	---
A9	INPUT	ACTIVE HIGH	Latched address	---
A10	INPUT	ACTIVE HIGH	Latched address	---
A11	INPUT	ACTIVE HIGH	Latched address	---
A21	INPUT	ACTIVE HIGH	Latched address	---
A3	INPUT	ACTIVE HIGH	Latched address	---
A14	INPUT	ACTIVE HIGH	Latched address	---
A15	INPUT	ACTIVE HIGH	Latched address	---
/RESET	INPUT	ACTIVE LOW	Reset pin	---
/CS510	OUTPUT	ACTIVE LOW	Enable the external UART	IE00H-IEFFH
/CE2	OUTPUT	ACTIVE LOW	Enable the NVSRAM	C000H-FFFFH
/BUSWIDTH	OUTPUT	ACTIVE LOW	Place processor in 8-bit mode	IE00H-IEFFH
OUT0	OUTPUT	ACTIVE HIGH	State counter bit 0	---
OUT1	OUTPUT	ACTIVE HIGH	State counter bit 1	---
/WAITE	OUTPUT	ACTIVE LOW	Hold 8XC196KC in a wait-state	---
OUT2	OUTPUT	ACTIVE HIGH	State counter bit 2	---
/CE0	OUTPUT	ACTIVE LOW	Enable both EPROM devices	2000H-BFFFH
/CE1	OUTPUT	ACTIVE LOW	Enable both SRAM devices	0000H-0FFFH 1000H-11FFH
INST	INPUT	ACTIVE HIGH	Select for memory overlap	---

TABLE 3.3 : GAL NO 1 PINOUT

Signal	Type		Function	Address
A1	INPUT	ACTIVE HIGH	Latched address	---
A2	INPUT	ACTIVE HIGH	Latched address	---
A3	INPUT	ACTIVE HIGH	Latched address	---
A9	INPUT	ACTIVE HIGH	Latched address	---
A10	INPUT	ACTIVE HIGH	Latched address	---
A11	INPUT	ACTIVE HIGH	Latched address	---
A12	INPUT	ACTIVE HIGH	Latched address	---
A13	INPUT	ACTIVE HIGH	Latched address	---
A14	INPUT	ACTIVE HIGH	Latched address	---
A15	INPUT	ACTIVE HIGH	Latched address	---
/RD	INPUT	ACTIVE LOW	Read signal issued by 80C196KC	---
/WR	INPUT	ACTIVE LOW	Write signal issued by 80C196KC	---
CELED	OUTPUT	ACTIVE HIGH	Enable the leds indicating pump active	1C06H
DIPSW16	OUTPUT	ACTIVE HIGH	Enable the 16-bit diagnostic switches	1C04H
CE1	OUTPUT	ACTIVE HIGH	Enable 12-bit counter no 1	1C06H
CE2	OUTPUT	ACTIVE HIGH	Enable 12-bit counter no 2	1C04H
CE3	OUTPUT	ACTIVE HIGH	Enable 12-bit counter no 3	1C02H
CE4	OUTPUT	ACTIVE HIGH	Enable 12-bit counter no 4	1C00H
CSRELAY	OUTPUT	ACTIVE HIGH	Enable the write to pump relays	1C00H
RESA	OUTPUT	ACTIVE HIGH	Reset counters 1, 2 or 3	1C08H
RESB	OUTPUT	ACTIVE HIGH	Reset counters 1, 2 or 3	1C0AH
RES4	OUTPUT	ACTIVE HIGH	Reset counter 4	1C0EH

TABLE 3.4 : GAL NO 2 PINOUT

3.1.2.2 *Wait-state implementation*

The Moore machine is used to implement a wait-state in the PHASE 1 hardware. The machine is coded in VHDL and programmed into a 22V10 GAL. The implemented machine offers a selection of up to six wait-states which are fed to the WAIT# output pin on the GAL. This WAIT# pin controls the READY input pin on the 196 microcontroller. The READY pin on the 196 microcontroller is an active high input pin which indicates that external memory is ready to

send or receive data. While the READY pin is held low, the bus controller inserts wait-states into the bus cycle. In order to implement the wait-states into the 196 microcontroller, the CCR (Chip Configuration Register) bits 4 and 5 in the Chip Configuration Register (CCR) of the microcontroller must be asserted to allow the READY signal control of wait-states.

A schematic diagram illustrating pin connections is included in Appendix L. A simulation displaying 1, 2 and 3 wait-states as required by the external components is presented in Figures 3.10 and 3.11.

FIGURE 3.10

FIGURE 3.11

The VHDL code listing for the state machine is presented in Appendix J-1. The state diagram for the Moore machine implemented is given in Figure 3.12.

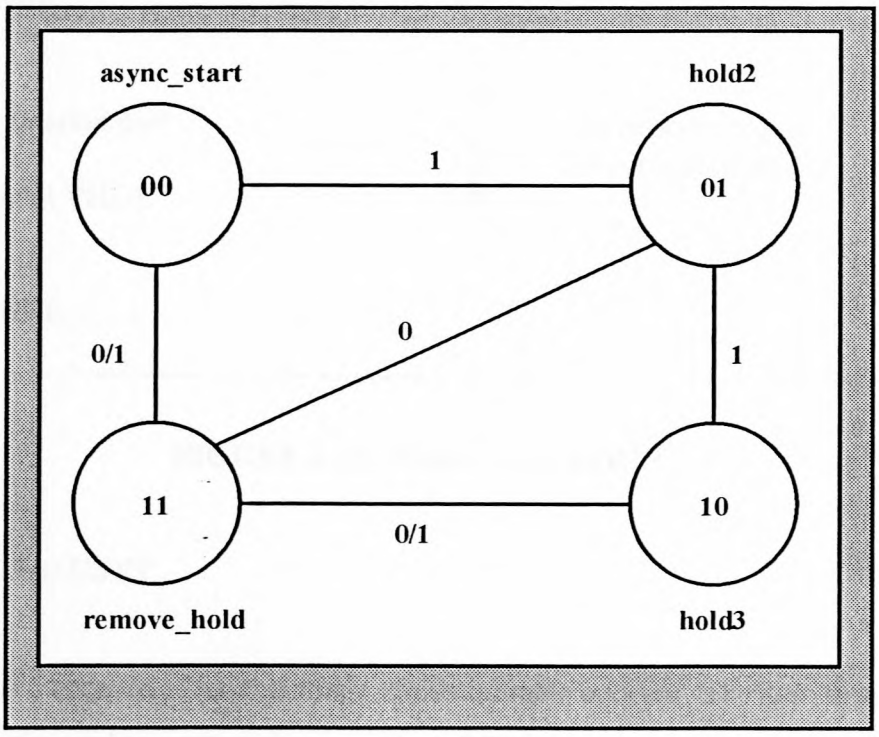


FIGURE 3.12 : STATE DIAGRAM OF MOORE MACHINE FOR WAIT-STATES

3.1.2.3 *PLD Programming*

The PLDs are programmed in VHDL since the language supports hierarchies, re-usable components, error management and verification. All of these features allow the design to be continually altered to support the components used in the design. The VHDL components are designed technology-independent, allowing libraries of existing components to be used, requiring little effort. The VHDL code is functionally verified with the WARP simulator.

The VHDL code used in the PHASE 1 GALs makes use of both concurrent and sequential data processing. Figure 3.13 illustrates the structural use of both constructs in this design.

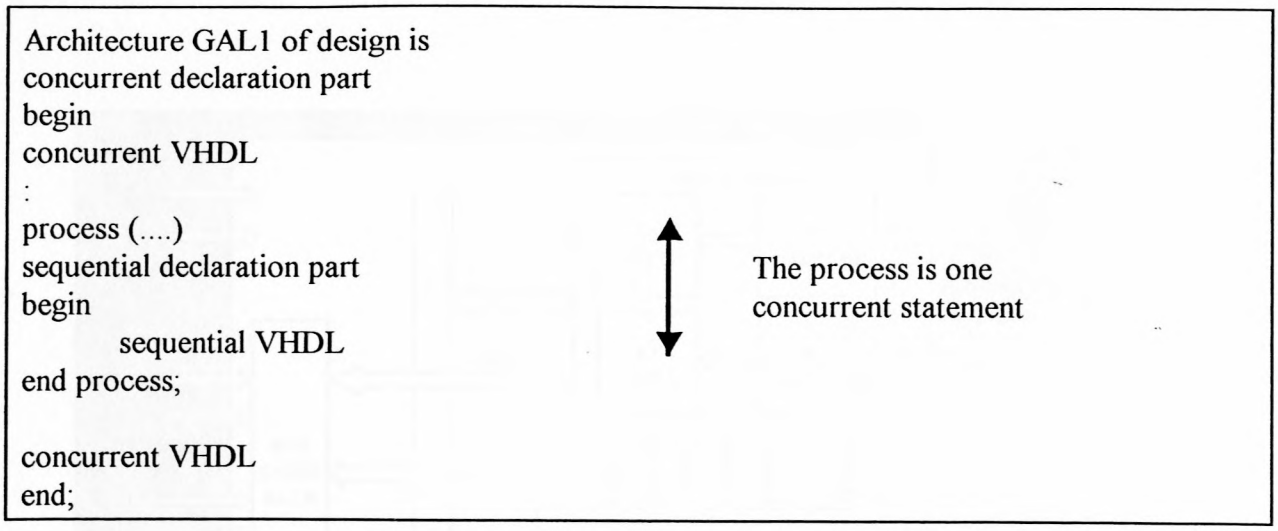


FIGURE 3.13 : VHDL CONSTRUCTS

3.1.3 *Intel 82510 UART*

The 82510 is used because of its vast range of configurable options. The 82510 has banks of registers which are used to control the operation of the device.

3.1.3.1 *Control Signals*

The 82510 has an I/O peripheral interface, a demultiplexed bus (8-bit data bus and 3 address lines), Interrupt, Read, Write, Chip Select and Reset pins which make up the system interface. A block diagram of the 82510 is presented in Figure 3.14.

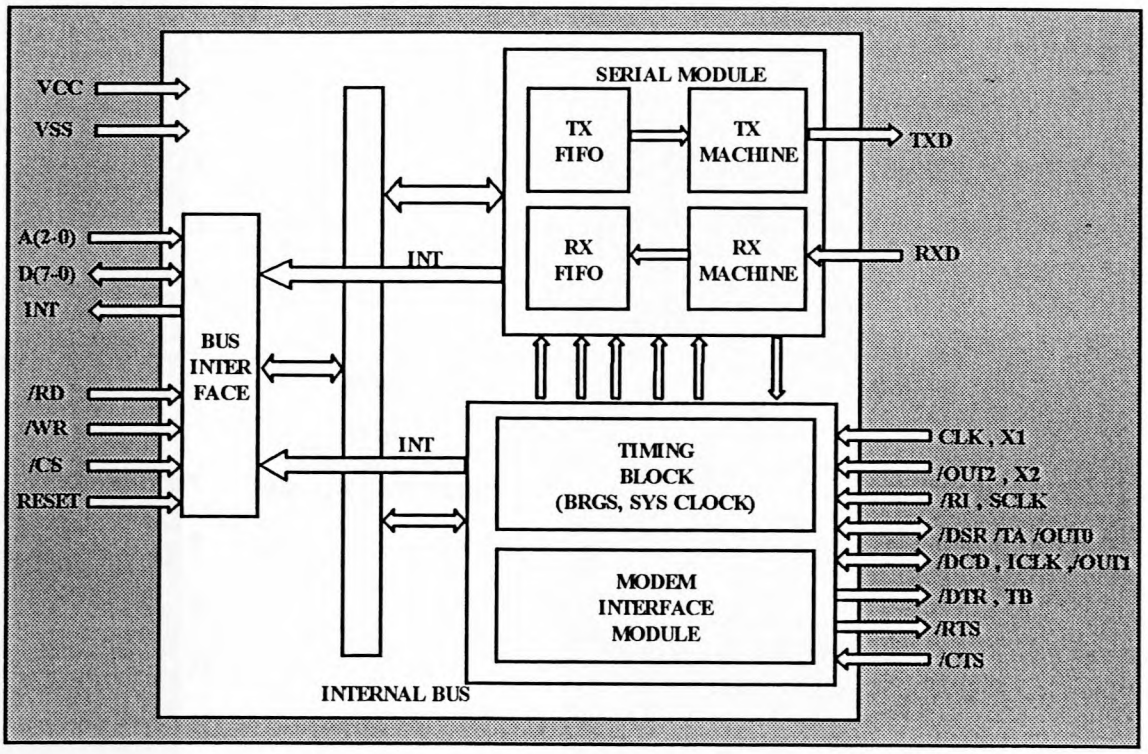


FIGURE 3.14

The four banks of registers provide support for the seven functional blocks shown in Figure 3.14. The registers allow configuration, provide status information about events/errors and may be used to send commands to each of the functional blocks. In this design, the 82510 is used as a communication interface to the Dallas touch memory. The two baud-rate generators/timers and on-chip crystal oscillator provide the functionality required for the Dallas touch memory interface. The register map for the UART is attached in Appendix D. The pin connections and schematic diagram for the UART can be found in Appendix L. The package outline can be seen in Figure 3.15.

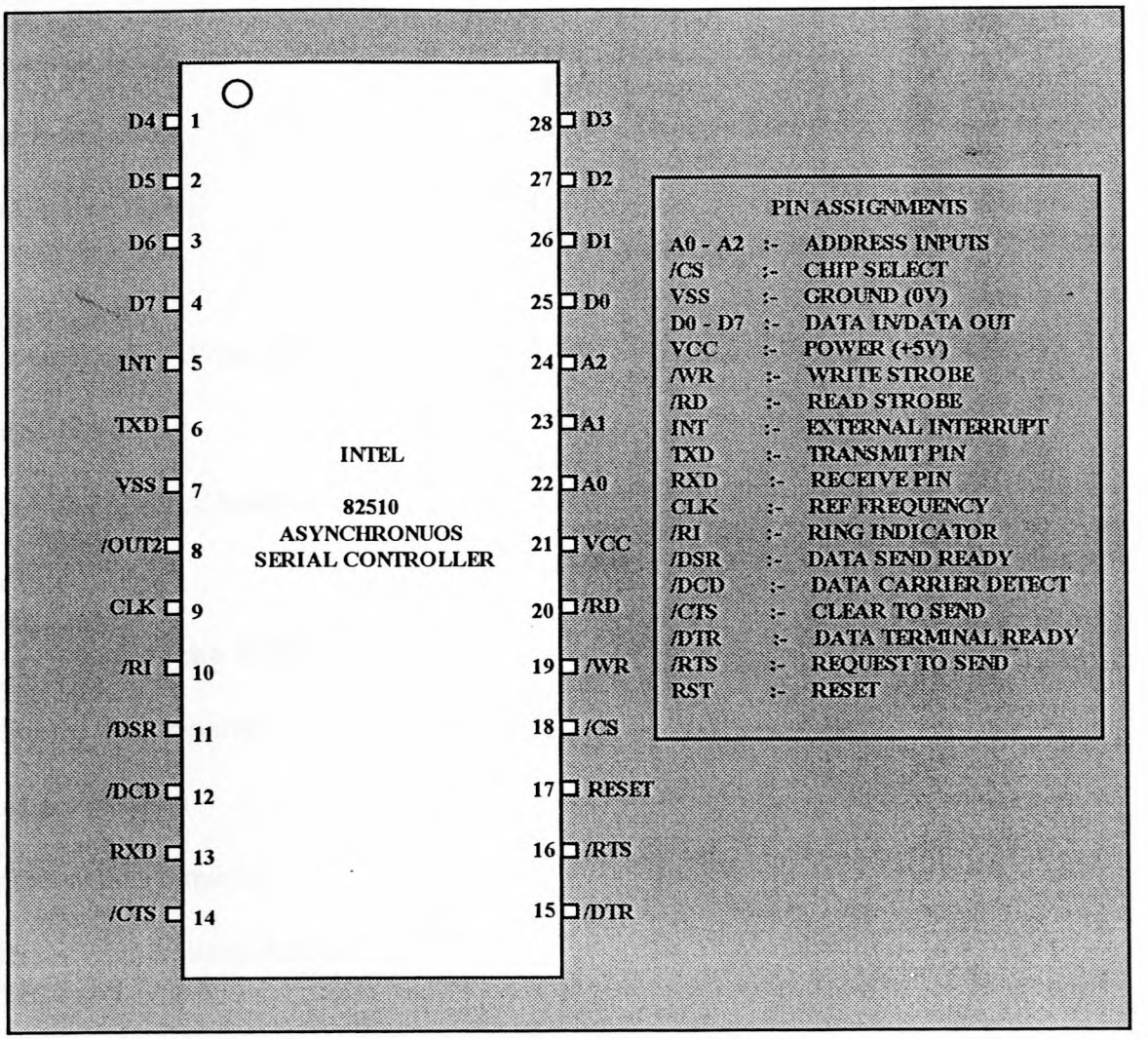


FIGURE 3.15 :

3.1.3.2 *Timing Considerations*

The timing equations for interfacing the 82510 with the microcontroller are as follows:

Read Cycle

Address to Read Low	=	$Tclcl - Tclav_{max} + Tclrl_{min} - Latch\ Delay_{max}$
Read Access Time	=	$2Tclcl - Tclrl_{max} - Tclvcl - Transceiver\ Delay_{max}$
Read Active Time	=	Trlrh
	=	2Tclcl - 46

Write Cycle

Address Valid to Write Active	=	$Tclcl + Tcvctr_{min} - Tclav_{max} - Latch\ Prop\ Delay_{max}$
Write Active Time	=	Twlwh
	=	2Tcld - 40
Data Valid to Write Inactive	=	$2Tclcl - Tcldv_{max} - Transceiver\ Delay_{max} + Tcvctv_{min}$

For this design

Ready Cycle

Read Access to Data Valid	=	
82510 Trldv	=	
additional time required	=	REFER TO APPENDIX 2A
	=	
Read active width	=	
82510 Trlrh	=	
additional time required	=	
	=	
Address Valid to Read Active	=	
82510 Tavrl	=	

This indicates that additional wait-states are required for this design.

3.1.4 *Non-volatile Memory/Real-time Clock*

A device is required for the retention of data acquired during system operation. In order to retain the data during power-loss, a non-volatile SRAM device had to be selected, but the requirement of a real-time clock led to the selection of a device with support for both real-time clock and NVSRAM. The DS1244Y Phantom clock manufactured by Dallas Semiconductor was chosen for this purpose. The package outline is presented in Figure 3.16.

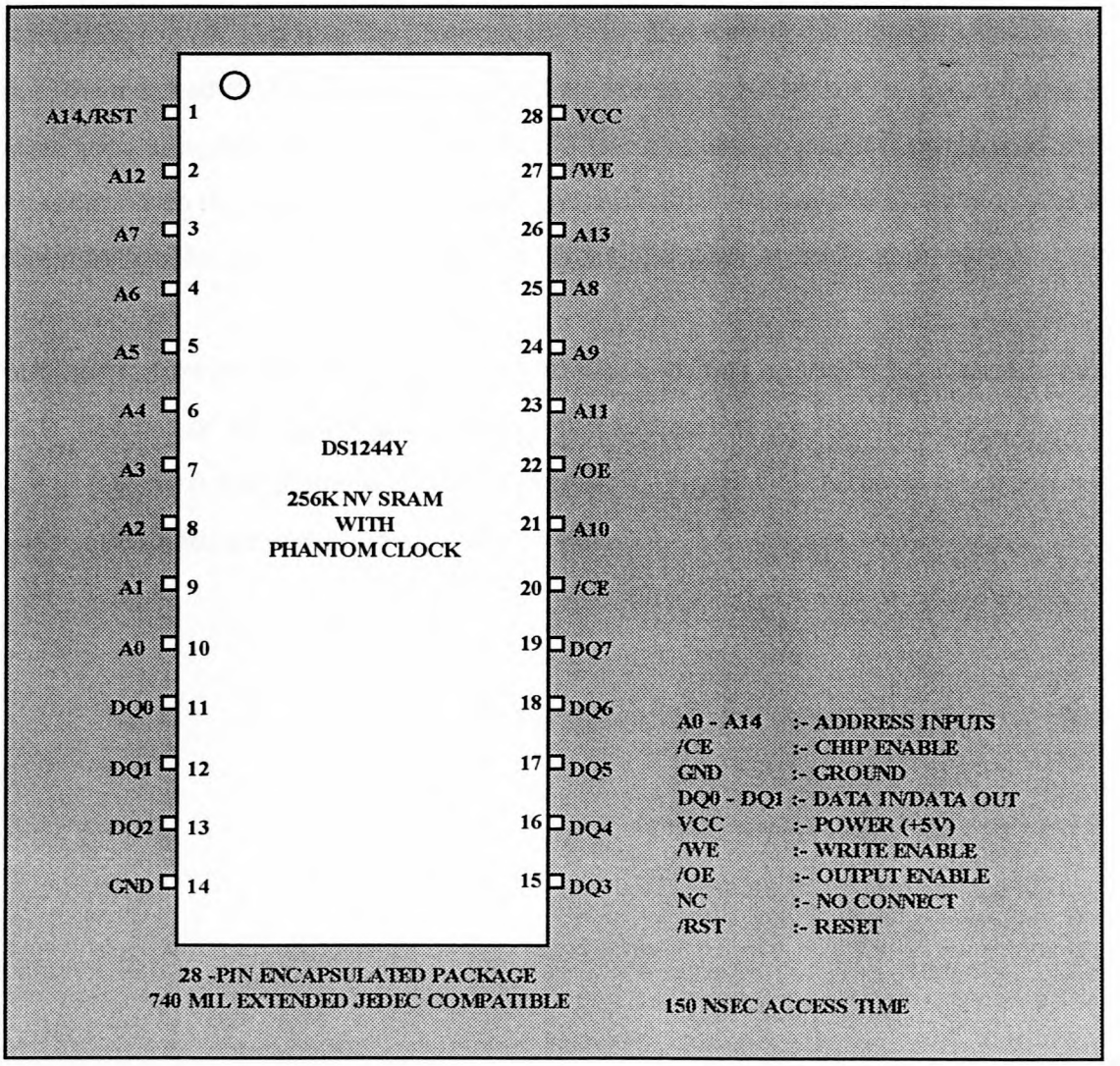


FIGURE 3.16 :

3.1.4.2 *Phantom Clock Operation*

The microcontroller communicates with the phantom clock once pattern recognition is established on a serial bit stream of 64 bits. This bit stream must be matched by executing 64 consecutive write cycles containing the proper data on DQ0 (data pin 0). All of the other chip accesses which occur before the recognition of the 64-bit pattern is directed to memory.

Directly after the recognition is established, the next 64 read or write cycles will either extract or update data in the phantom clock chip. During this time, memory access is inhibited. The write cycles generated to gain access to the phantom clock are also writing data into the memory location in the matched RAM. To avoid overwriting data in the RAM, one address location in the RAM is set aside as a phantom clock scratch pad. Once a bit stream match is established, the pointer increments to the next location and waits for additional write cycles. If a read cycle is encountered before the full 64 bits of clock data is read, the clock access is aborted.

The clock information is contained in 8 registers of 8 bits, making up the 64 bit stream containing clock data. Reading or writing registers, requires that all registers are read or written, starting with bit 0 of register 0 and ending with bit 7 of register 7. Figure 3.17 presents the phantom clock register definition as programmed in this project.

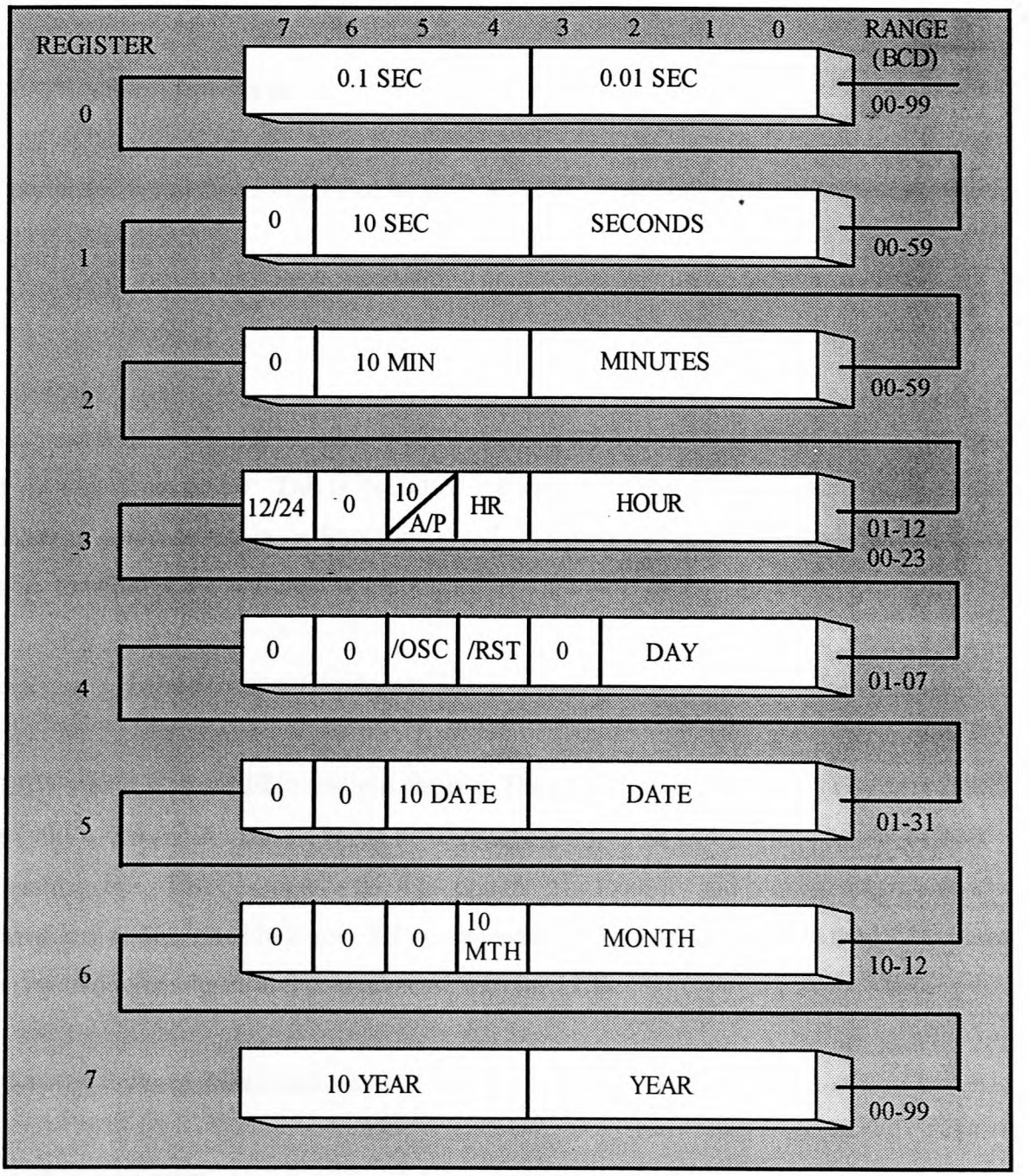


FIGURE 3.17 :

The control sheet used to format the data required for updating the phantom clock is presented in Appendix E.

3.1.5 *Peripheral Interfaces*

The peripheral components interfaced to the 80C196KC microcontroller are discussed in more detail in this section.

3.1.5.1 *Keypad*

The keypad is the main user interface with its columns driven via 4 bits from port 1 and the rows read via 4 bits from port 0. This is illustrated in Figure 3.7. The driving capacity of 20 mA is sufficient to obtain secure input from the keypad. A debounce routine is included in the software in order to improve the key press operation.

3.1.5.2 *Liquid Crystal Display (LCD)*

the LCD used is a 16x4 LED backlight display. The LCD control is performed by port 2 bit 6 and port 1 bits 4, 5,6 and 7. The LCD can be interfaced with a 4 bit address/data bus or an 8 bit address/data bus. This design uses the 4 bit option. The intensity and backlighting is controlled via hardware as illustrated in Figure 3.7 or Appendix L. Appendix F presents the LCD control data required to configure and communicate with the LCD.

3.1.5.3 *Pump Status Indicators*

A set of tri-color LEDs are used to provide status indication for the pumps controlled by the system. The hardware design allows the LEDs to be forced into one of the colors indicating unused pumps. This implies that the system designer has the option to drive any LED colors as required. The LEDs are driven via the ULN2803A darlington pair drivers. The output current from the ULN2803A is controlled via in-line resistors to control LED intensities.

3.1.5.4 *Input/Output Signals for Pump Control*

The microcontroller-based system monitors the status of the pump nossels. This is accomplished by feeding the input signals through Schmidt triggers and into the 4 input pins of port 0. This is illustrated in figure 3.7. The pumps are enabled by the microcontroller via four interpolar relays. The relays are 47 Watt, 5 Volt PCB mounted relays. The interpolar relays are driven by a ULN2803A darlington driver. The darlingtons are switched via tri-state latches of the address/data bus of the system.

The four asynchronous pulse counter outputs are fed onto the microcontroller data bus via Schmidt triggers and 74LS541 tri-state latches. A buzzer is also driven via an interpolar relay. The buzzer is used to provide an audible indicator for user confirmation of Dallas tag or keypad input.

3.1.5.5 *System Diagnostics*

A diagnostic mode is entered on power-up of the system if any of the DIP switches in the 16 bit bank are asserted. The diagnostics modes vary according to the state of the bank of DIP switches. The diagnostics program test the following:

- NVSRAM read and write cycles
- 16-bit SRAM read and write cycles
- all pump status indicators
- LCD initialization and control
- all relay driven outputs
- all keys on the keypad
- data integrity via the communication link
- Dallas tag communication
- counter inputs

3.1.5.6 *Dallas Tag Communication Interface*

The hardware configuration for the Dallas touch memory requires a 1-wire bus consisting of a single line. Due to the feature that multiple tags can be connected to the bus simultaneously, each device must have an open drain connection or a tri-state output. The DS1996 used in this design has a open drain configuration as shown in Figure 3.18.

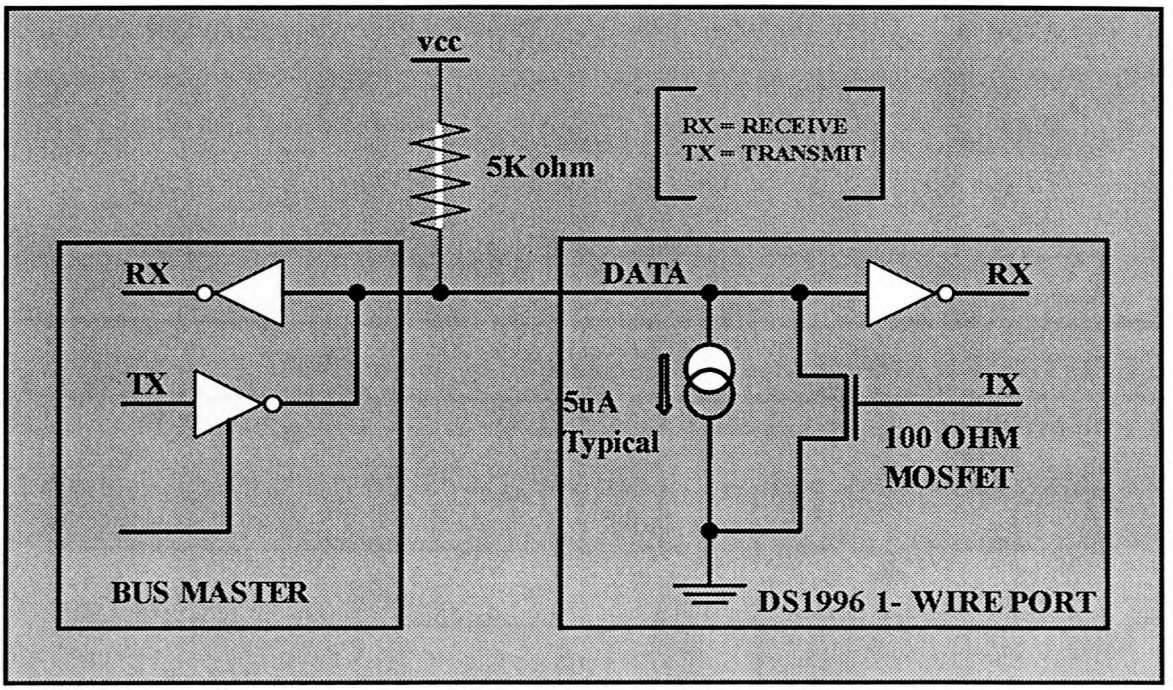


FIGURE 3.18 :

At the regular speed, the bus has a maximum data rate of 16,3 k bits/second. The speed is boosted to 142 k bits/second by activating the overdrive mode. A pull-up resistor of 5 kΩ is required by the 1-wire bus. All of the transactions on the 1-wire bus begin with an initialization sequence. This sequence consists of a reset pulse transmitted by the bus master which is followed by a presence bus transmitted by the slave (tag). The reset pulse width (t_{RSTL}) is 480 μs minimum for regular speed operation and 48 μs minimum for overdrive speed operation. The timing diagrams can be found in [64].

The schematic for the hardware developed for communication with the Dallas touch memory is presented in Figure 3.19.

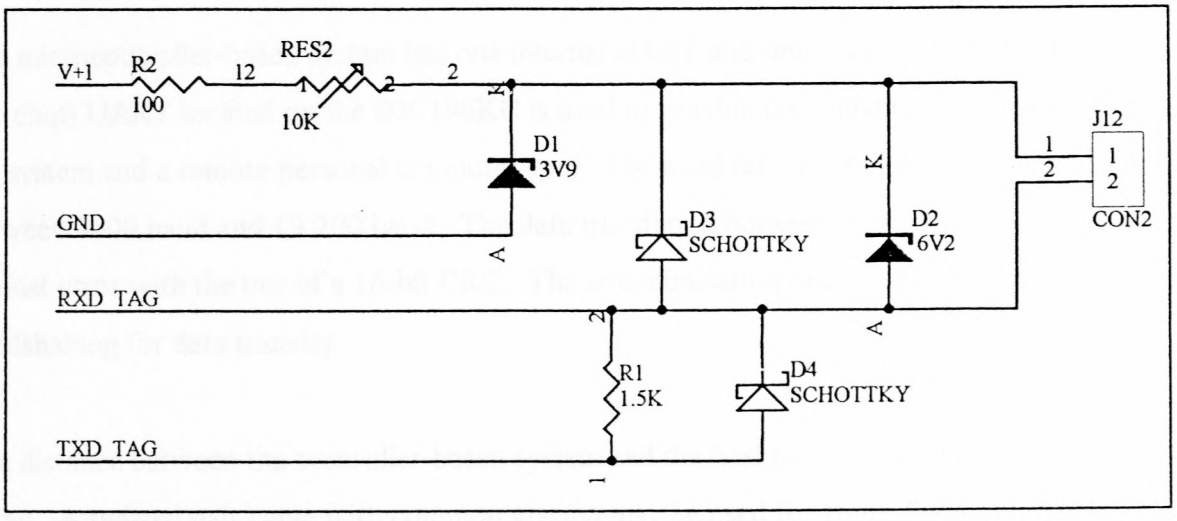


FIGURE 3.19

The same hardware is used for communication with a pc and the microcontroller board, since it is interfaced via the RS232 serial communication port. The Dallas touch memory devices used in this project are listed below with their functions:

DS1991 : used as either a driver or vehicle identification and interrogation
DS1994 : used as a system configuration and real-time clock tag
DS1996 : used to transfer data blocks form remote system to pc

The memory maps for the DS1991, DS1994 and DS1996 as used in this design, are presented in Appendix Z-1, Z-2 and Z-3.

3.1.5.7 *RS232 Communication Support Hardware*

The microcontroller-based system has one internal UART and one external UART. The internal (on-chip) UART located on the 80C196KC is used to provide the communication link between the system and a remote personal computer (pc). The baud rate for the serial link can be adjusted between 9600 baud and 19 200 baud. The data transferred between the two systems is tested against error with the use of a 16-bit CRC. The communication makes use of software handshaking for data transfer.

The distance between the controller-based system and the host pc may vary between 5m and 100m. A twisted pair cable with common ground may be used for short distances, but the attenuation is too great over larger distances. Because of this factor and the danger of lightning strikes damaging the equipment, optic fiber or radio communication provides a safer alternative.

All three options are presented:

- *Optic Fiber*

Both the plastic and glass fiber provide protection against voltage spikes, EMI and radio frequencies, but glass fiber with its associated components, is up to 10 times more expensive than plastic fiber. The glass fiber connections use laser diodes or high-end LEDs as a light source, while plastic fiber uses simple, inexpensive and widely available LEDs and photo diodes. The plastic fiber used in this design is both rugged and inexpensive. The optical fiber is composed of a

polymethylmetha-crylate (PMMA) core which is encased in fluoride-based carbon polymer. The cable has a diameter of approximately 2,2mm when coated.

The plastic fiber, when used over short distances with allowances for attenuation, can be effectively cleared and finished with a heated blade or hot plate. Recent experiments with this plastic fiber have demonstrated speeds in the gigabit per second range. Different wavelengths have varying attenuation characteristics and must be selected based on the application. The emitter and detector used in this design operate in the 650nm range of red visible light and is suitable for optical transmission at up to 100m. It is important to note that the maximum data rate is heavily influenced by the following:

- distance
- attenuation of the fiber
- loss due to connections
- sensitivity of the detector

The fiber used in this design is relatively inexpensive, and supports a baud rate of up to 19 200 for minimal error transmissions. A board converting the RS232 to optic pulses had to be designed. The completed board is displayed in Figure 3.20. The schematic PCB and bill of materials of the conversion board are presented in Appendix H.

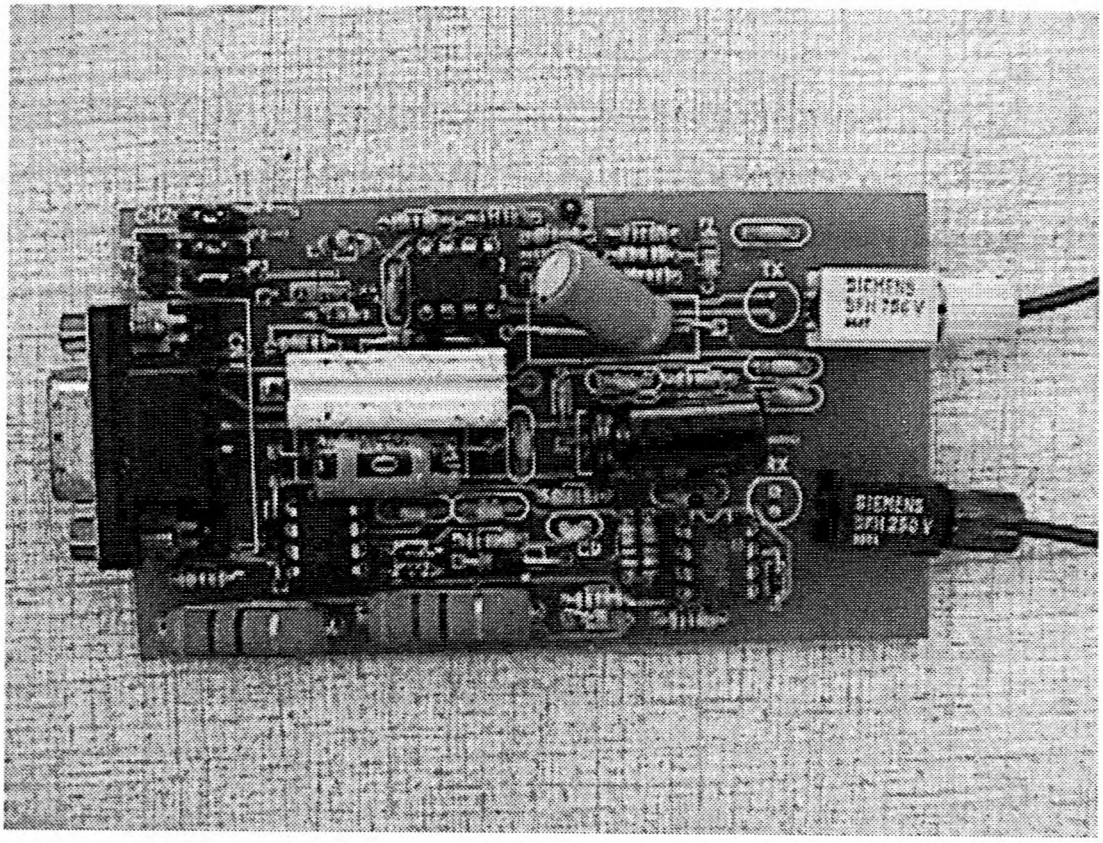


FIGURE 3.20 : PHOTOGRAPH OF RS232 TO 1-WIRE COMMUNICATION SYSTEM

The design use a LM311 to amplify the transmitted signal from the TXIN pin of the RS232 port sufficiently to drive the transmitter diode. The transmitter diode used has a wavelength of 650nm which places it in the visible range. This supports visual inspection for diagnostic purposes. With the correct feedback design and set references, the incoming light pulses detected by the photo diode are fed through a precision amplifier CA314DE once offset against a common reference. This output can be monitored via a monitor pin. This output is then passed through another LM311 amplifier which is then fed to the RXOUT pin of the RS232 port. These connections are shown in the schematic in Appendix H.

– *Radio Link*

The radio transmission is accomplished with the use of a transceiver module manufactured by Radiometrix. Only the peripheral hardware required for the conversion of RS232 to the interface format provided by the module had to be designed. Transmission occurs at a frequency of 433 MHz and the band is prepaid by the manufacturer. The range is limited to 120m outside of buildings by the module design. One module is connected to the microcontroller-based system and the other to the remote pc.

3.1.6 *Supervisory Control Circuitry*

As discussed in section 3.1.4 the NVSRAM used to store the data is the DS1244Y IC write protects itself for any voltage below 4.25 V. This is however insufficient and test conditions yielded results to support this. Data corruption still occurs under conditions where data is currently being processed in the event of a power failure. In order to overcome this problem, additional hardware is utilized.

Figure 3.21 represents the system which uses both the DS1233 10% and DS1233 5%, 5 Volt econo-reset semiconductor chips manufactured by Dallas Semiconductor.

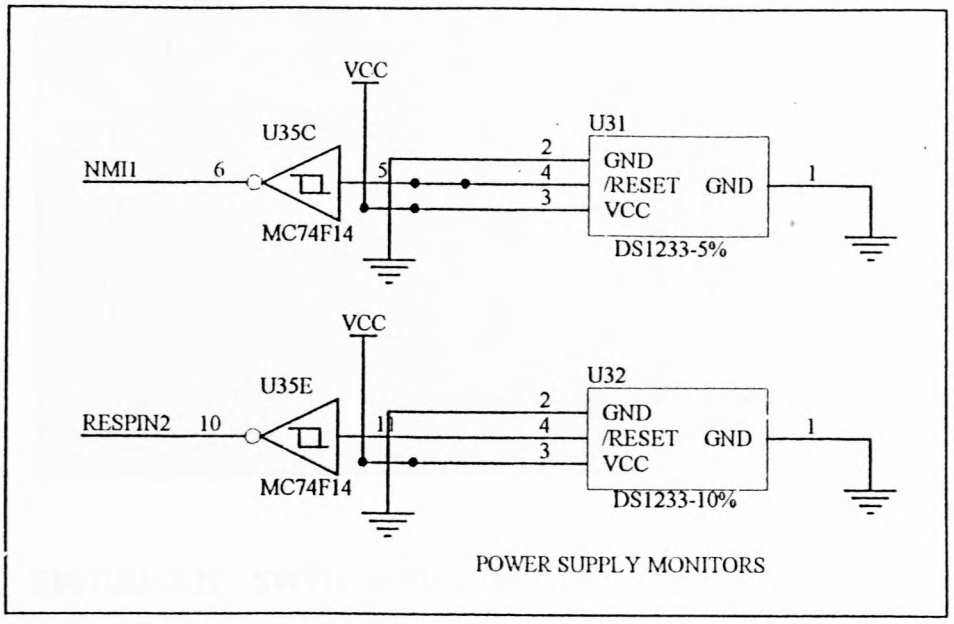


FIGURE 3.21 :

In the event of a 5% voltage drop from V_{CC}, the DS1233 5% component issues a signal connected to the non-maskable interrupt service routine of the 196 microcontroller. This service routine withdraws all of the data and addresses from the busses and places the microcontroller into an idle state. If the supply voltage rises again without falling below 10% of V_{CC}, then the microcontroller will time-out of the interrupt service routine after a set interval to resume with processing.

If the supply voltage however continues to fall below 10% of V_{CC}, then the DS1233 10% will issue a signal resulting in a total reset of the microcontroller based system.

3.1.7 *Electromagnetic Interference Reduction Techniques*

The Electromagnetic Interferences (EMI) are the electrical noises other than those inherent in the circuit components. Supply line transients are EMIs resulting from the switching of heavy loads onto off AC or DC power lines. Figure 3.22 shows a typical circuit presenting these transients.

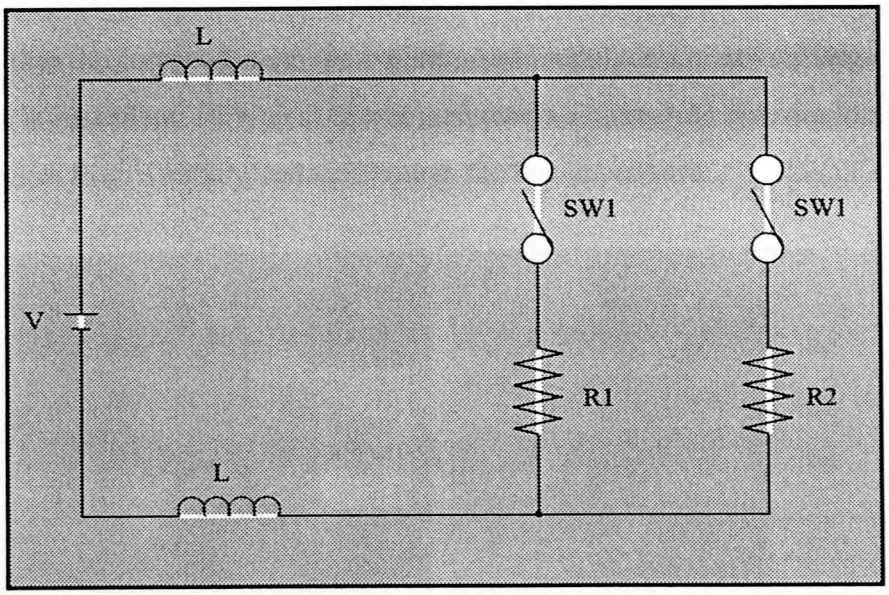


FIGURE 3.22 : SWITCHING CIRCUIT WITH TRANSIENTS

If both loads R_1 and R_2 draw current, the line current flowing through the line inductance establishes a magnetic field of some value and when any one of the loads are switched off, the inductor generates a voltage transient called an "inductive kick". This is worse when the switch is opened. The design must should be modified to avoid this.

Noise in ground lines which are referred to as ground loops should also be avoided. When two earth-grounds are at different potentials, an unwanted current is established in the wire. Radio Frequency Interference (RFI) can cause noise in the form of electromagnetic radiation. To overcome this noise, effective shielding techniques are required. In the ground loop, the area of the loop is what matters. If the feed and return paths are parallel to each other, the magnetic fields will cancel out, thus this is the best solution to the problem and where possible is implemented in this design.

The best way to minimize loop areas is to use a ground plane. This may be expensive and an alternative is to use the gridded-ground PCB layout. This technique is implemented in this design. Ground traces are laid out between signal traces where possible. This is done with vertical traces on one side of the PCB and horizontal traces on the other side. These must be connected by a through feed. The hardware (relay) and signal grounds are kept separate to avoid interference of noise on the signals.

Due to the fact that the current loop in the typical power supply is a relatively large one, a fair amount of noise is picked up. A typical circuit is shown in Figure 3.23 where a load is trying to draw current spikes from a supply voltage through the line impedance.

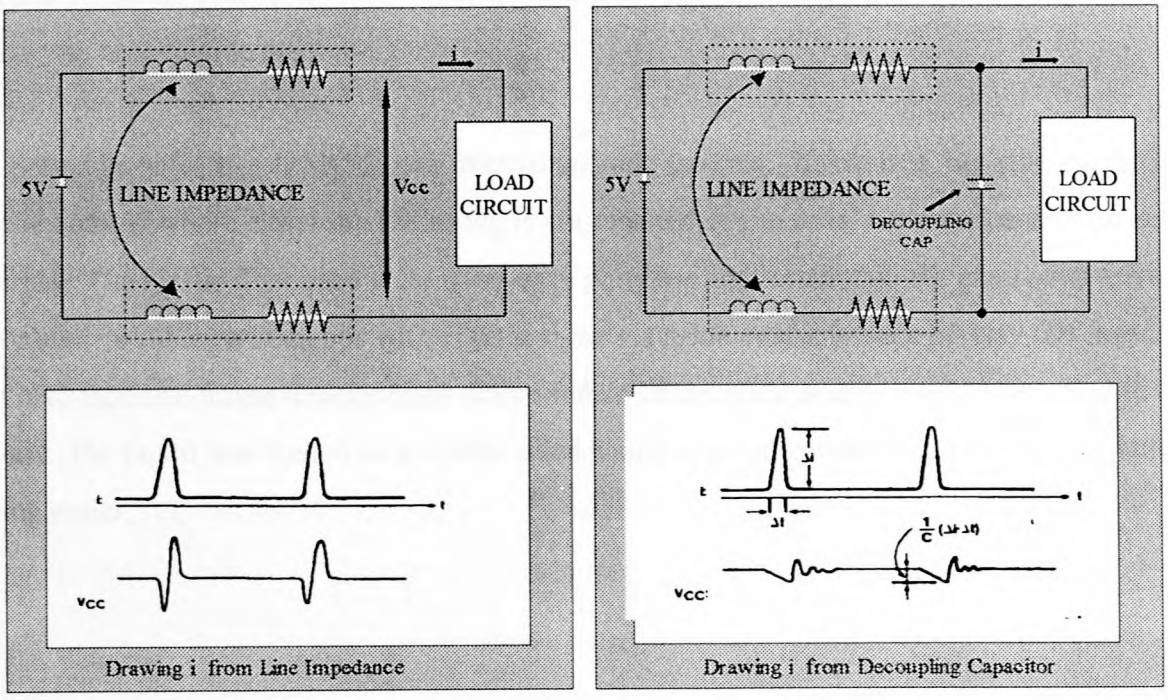


FIGURE 3.23 : DECOUPLING CAPACITORS IN SWITCHING CIRCUITS

A decoupling capacitor is added to the circuit as shown in order to supply the current spikes through a smaller line impedance. The temptation to add a resistor or inductor for a filter should be avoided since it would lead to slowing down the speed which the decoupling cap is refreshed [19]. The two types of decoupling caps used in this design are the board decoupling capacitors and the chip decoupling capacitors. The board decoupling caps are normally 10 to 100 μF electrolytic caps placed near where the power supply enters the board. These caps are to refresh the charge of the chip decoupling caps. The chip decoupling caps are ceramic and are normally 0,1 to 1 μF placed as near to the chip as possible.

3.1.8 *Cost-effective Design Methodology*

The microcontroller system has five memory sockets. The sockets can be populated with either ROM or RAM. The RAM and ROM can be configured for both 8 or 16-bit operation with the aid of jumper settings. The 80C196KC also has support for 16 K of on-chip ROM. This implies that the minimum external memory may be one 8-bit RAM chip. The /EA pin on the 80C196KC microcontroller sets the ROM as internal or external memory. The real-time clock required by the system is provided on a 32 K byte phantom clock which is a non-volatile RAM chip with a clock built into it. The JEDEC pinout of the DIP package also supports the use of a smart socket supplied by Dallas Semiconductor. This socket has a battery built into it for converting volatile memory to non-volatile memory.

The design includes two UARTS, one internal and one external. If required, the external UART may be omitted where either the Dallas tag is not required or the serial communication link is not required. The plastic fiber used is far cheaper to purchase and install than the glass alternative. The complete Address/Data bus with control signals is made available via a 60 way IDC header. This is to facilitate future development where additional interface boards need to be added to the system. The board was routed as a double-sided board in order to make prototyping and fault-finding easier.

3.2 Software Design

3.2.1 *Microcontroller Program*

The BSO tasking assembler and C compiler was used to compile the source code for the 80C196KC. A flow diagram is presented with the source code in Appendix I.

3.2.2 *Generic Array Logic Program*

A GAL (generic array program) is used to perform the function of controlling access to the external devices connected to the 80C196KC microcontroller. The VHDL code programmed into the GAL has a static component and a sequential logic component. The chip select control signals are generated using the static logic component while the state machine implementation is done in the sequential component. The high and low write control signals (/WRH, /WRL) are also generated with the help of the BUSWIDTH signal issued by the GAL. The VHDL code was compiled, synthesized and simulated using the 'WARP' software package supported by CYPRUS.

A screen plot from a logic analyzer presenting the wait-states required by the external UART, 16-bit EPROMS, 8-bit NVRAM and the 12-bit counters are presented in appendix T.

The results from the synthesis, simulation and software analysis software used (WARP) is presented in Appendix J-3. The software simulation is verified with the aid of a hardware logic analyzer. These waveforms are presented in Appendix K.

3.2.3 *PC Based Program for Serial Communication Link*

The software code written in visual basic is included on the attached CD, with only the windows forms and operation brochure presented in Appendix K.

3.3 Manufacture and Implementation of PHASE 1 System

3.3.1 *Printed Circuit Board Design*

The system design was entered into the ACCEL Tango schematic package located on a pc with the DOS 6 operating system. A netlist file was then generated from the completed schematic. The footprints for the required components were placed on the board outline in the ACCEL Tango PCB package. The netlist was the loaded into the PCB software and the board was routed according to the required setting using the ACCEL Tango Router package. Further modifications were made to the PCB to reduce the possible electromagnetic interferences (EMI). The schematics and PCBs of the PHASE 1 system are presented in Appendix L.

3.3.2 *Power Supply*

A switch-mode power supply is used to provide power to the system. The design of the power supply incorporates an "off-the-shelf" device since it was found that the development of a similar device would have proven far more expensive. The power supply is a standard 250 Watt personal computer power supply. The microcontroller board has three transorbs, 5V, 12V and -12V devices placed across the supplies. The transorbs are used mainly for transient suppression and protection against reverse polarity. The non-volatile memory is backed by a 10 year guaranteed battery to assist in the avoidance of memory loss. Capacitors are used in the circuit to suppress further voltage transients created by switching external inductive loads.

3.3.3 *System Installation*

The system is installed in a rigid state structure with the external connections as illustrated in Figure 3.26.

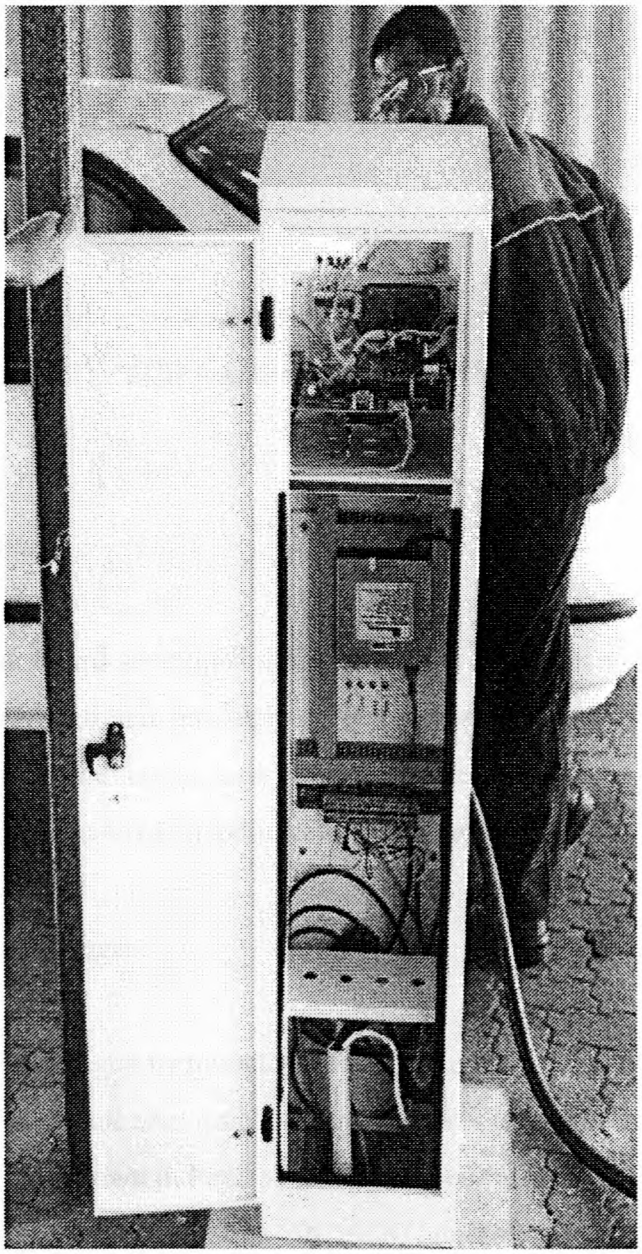


FIGURE 3.26 :

3.4 Commissioning and Field Tests of PHASE 1 System

On completion of the system installation a comprehensive diagnostic sequence is followed. This sequence tests all of the peripheral components, the memory block, Dallas tag communication and the serial communication link.

3.4.1 *Problems and Solutions*

3.4.1.1 *System Corrosion*

The PCBs and wire connectors developed corrosion which may eventually have led to increased noise and poor conduction. The PCB was then coated and the connectors cleaned with contact cleaner and sealed.

3.4.1.2 *Overheating*

The main microcontroller board developed timing problems due to an increase in temperature brought about by poor ventilation in and high ambient temperatures. Some of the component grades were altered to accommodate greater temperature fluctuations. A small fan with additional ventilation openings were introduced to solve the problem.

3.4.1.3 *Transient Voltages*

The system was severely damaged by inductive-kicks produced when external relays were switched under load. The circuit responsible for the transient is similar to that of Figure 3.24. A voltage spike similar to that shown in Figure 3.24 was superimposed on the 15 V supply. This disturbance resulted in the destruction of the 80C196KC microcontroller. A 0,1 μF 250 V decoupling capacitor was installed to irradicate the problem. The solution proved successful.

3.5 Recommendations for Improvements to the PHASE 1 System

It is recommended that the component count be reduced on the board in order to minimize possible fault occurrences and component costs. The power consumption needs to be reduced in order to reduce the fault possibilities. Greater configurability needs to be offered by the system in order to provide a more versatile system. Improved peripheral support is required for the addition of larger control systems.

Hardware offering "plug-in" replacements for a range of gate densities is a necessity. The replacement of most gate-level components with a CPLD, would reduce the EMI of the PCB and provide most of the recommendations mentioned above.

It would also allow for a reduction of fault diagnosis and repair time. Any future upgrades may then be simulated prior to implementation, reducing development time. The inclusion of further transient suppression techniques is recommended with the use of in-line fuses.

Chapter 4

Design of a PHASE 2 Microcontroller-based System using Complex Programmable Logic Device Technology

4.1 PHASE 2 Hardware Design

The new PHASE 2 design incorporating the two FLEX10K CPLDs is presented in figure 4.1.

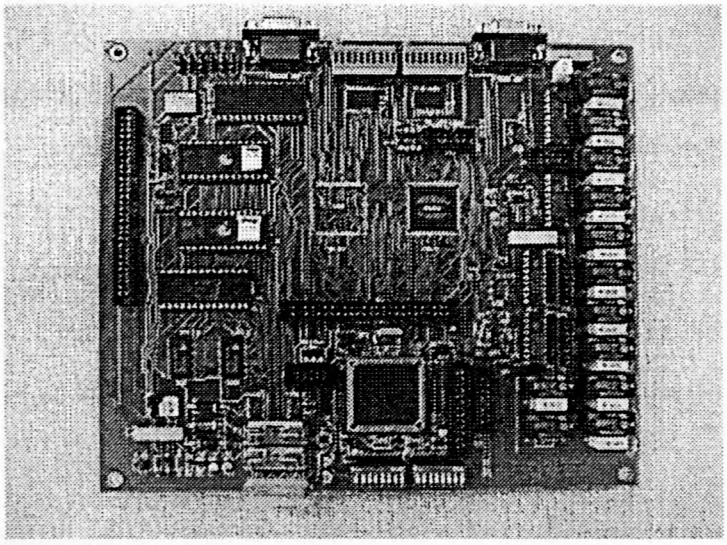


FIGURE 4.1 : PHOTOGRAPH OF THE PHASE 2 CPLD INCORPORATED DESIGN

All of the conclusions and recommendations drawn from the PHASE 1 design are implemented in the PHASE 2 design in order to produce an improved product.

4.2 Alterations to PHASE 1 Hardware Design

In order to implement the necessary recommendations made on the PHASE 1 design, the following improvement were made.

4.2.1 *Memory Map*

The original memory map of the 80C196KC microcontroller system only provided for a 64 K memory block whereas the new PHASE 2 design makes use of the INST pin on the microcontroller. This supports ROM and RAM overlap by differentiating between instructions and data. The greater gate density offered by the use of CPLD allows for the implementation of support for eight pump control instead of four. Appendix C-2 presents the new peripheral I/O addresses.

4.2.2 *Supervisory Control Circuitry*

Figure 4.2 represents the circuit configuration implemented for the second system.

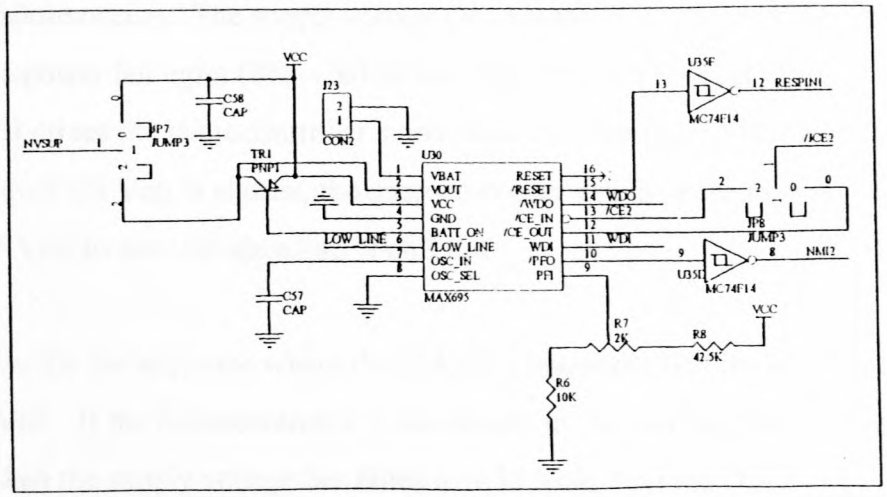


FIGURE 4.2

This system uses the MAX 695 chip manufactured by MAXIM. This supervisory circuit is designed to provide the following:

- an adjustable low voltage reset signal
- a non-maskable interrupt issued as power failure warning
- RAM write protection by controlling the chip select line of the memory
- an adjustable watchdog timer to verify software execution
- a battery-backed power supply voltage
- automatic battery switch-over on adjustable variation between battery voltage and V_{CC}

As can be seen from Figure 4.2, the NVSRAM is powered from V_{OUT}. V_{OUT} is the voltage output which is internally connected to V_{CC} when 5 Volt power is present, or to the battery supply (V_{BAT}) when V_{CC} is less than the battery voltage. V_{OUT} is capable of supplying up to 50 mA. The additional current is provided via the external pnp transistor.

A voltage detector monitors V_{CC} and generates a reset output to hold the microcontroller's reset line low when V_{CC} is below 4,65 Volt. An internal monostable holds the reset line low for 50 milliseconds after V_{CC} rises above 4,65 Volt. This prevents the repeated toggling of the microcontroller's reset line. The MAX695 issues a non-maskable interrupt to the microcontroller when a power failure occurs. The supply voltage (V_{CC}) is monitored via two external resistors connected to the power fail input (PFI). When the voltage at PFI falls below 1,3 Volt, the power fail output (PFO) drives the microcontroller's non-maskable interrupt (NMI) low. If a power fail threshold voltage of 4,8 Volt is chosen, the microcontroller will have the time when V_{CC} falls from 4,8 Volt to 4,65 Volt to save the data into NVSRAM.

The design allows for the sequence where the MAX695 issues the first interrupt as a power fail warning at 4,8 Volt. If the microcontroller is not already in the non-maskable interrupt service routine (ISR) when the supply voltage has fallen to 4,75 Volt, then the DS1233 5% will initiate the interrupt and force the microcontroller into the (ISR). From this it can be seen that the

DS1233 5% is the backup for the MAX695 for initiating the NMI. The microcontroller reset line is pulled low either by the MAX695 or the DS1233 10% at 4,65 Volt after the necessary "housekeeping" has been performed on the system. This is done to avoid data corruption in the two CPLDs which are programmed via an EPC 1441 serial EPROM. The system is designed to reconfigure the two CPLDs on a system reset. A backup is provided irrespective of the sequence of reset line initiators.

The chip select issued by CPLD 1 (CE IN) is fed into the MAX695 which has a chip select output line (CE OUT) which drives the chip select inputs of the memory. The signal CE OUT follows CE IN as long as V_{CC} is above the 4,65 Volt reset threshold. If V_{CC} falls below the reset threshold, the signal CE OUT goes high, independent of the logic level at CE IN. This prevents the microcontroller from writing erroneous data into RAM during power-up, power-down, brown-outs and momentary power interruptions. The low line output is used to indicate when V_{CC} falls below 4,65 Volt on the board.

The MAX695 has a watchdog timer which is used to verify that software execution is being performed by the microcontroller. This timer must be reset after an adjustable time interval and a failure to do this would result in a system reset. The system reset would be initiated by the watchdog output (WDO) signal driving the reset line low on the microcontroller. The address decoding for the watchdog timer input pin (WDI) is performed by the CPLD in order to ensure that it is also still functional. A system reset would result in the CPLD also being reconfigured.

When considering the timing at which the supply voltage would fall, we assume a reasonable time of 100 microseconds for the voltage to fall from 5% of V_{CC} to 10% of V_{CC}. The time chosen is based on actual measurements.

The microcontroller in this project runs at a clock speed of 20 MHz. Since the microcontroller is a 16-bit device, requiring on the order of approximately 12 clock cycles (6 state times) to execute a single instruction, it is possible to calculate the average number of instructions that the controller would be able to execute between the 5% and 10% trip points.

$$\frac{1}{20\,MHz} \text{ x } 2 = 100 \text{ nano seconds} = 1 \text{ state clock cycle}$$

at an average of 6 state times per instruction = 6 clock cycles/instruction

With a total time of 100μs it can be seen that a total of 166 instructions can be executed during a power-down prior to the NVSRAM write protecting itself or the system resetting. If required, the rate of fall of a voltage of a power supply during a power failure can be slowed by adding capacitance.

4.2.3 *Improved Configurability*

The PHASE 2 design has support for 2 FLEX 10K CPLDs. The system control signals are implemented in the first CPLD with peripheral support for four pumps. The second CPLD need only be introduced onto the PCB if additional support for a further four pumps is required. The external UART can also be implemented in the second CPLD as a megafunction if required. The second CPLD also has some of it's spare output/input pins brought out to a IDC connector.

Since both CPLDs have the system bus passing through them, internal memory can also be implemented with support for most megafunctions. Any of the timing alterations which need to be performed on the system can be done inside any of the two CPLDs. The pin configurations for the five 28 pin JEDEC sockets on the PCB are carried out in the first of the two CPLDs. The configurability allows for various capacities of ROM or RAM chips to be used in any of the sockets.

4.2.4 *CPLD Inclusion*

The use of CPLDs in the new design allows for a reduced component count which in turn reduces problematic component sourcing and faults due to incorrect component population. This reduction in component count allows the new design to be implemented on a single PCB. The inclusion of megafunctions into the design leads to improved system performance and eliminates the need for hardware components where megafunctions can be used to implement a substitute. The external UART can be substituted with a megafunction reducing component count still further.

The feature of in-circuit reconfiguration of the FLEX range of CPLDs also supports rapid prototyping allowing for immediate upgrades to be simulated, synthesized and analyzed and finally implemented.

4.2.5 *JTAG and BST Implementation*

The JTAG and BST systems can be implemented into the system in order to facilitate bare-board testing of PCB and therefore produce reduced production times. This has not been included in the thesis due to the time constraints experienced.

4.2.6 *Peripheral Upgrades*

The Dallas tag communication circuit has been included in the main PCB. The LCD support circuitry has been included in the new PCB. Dedicated keypad and LCD headers have been incorporated into the new design. The new PCB incorporates support for twelve interpolar relays for switching external devices.

4.2.7 *Flexible Timing Design*

The PHASE 2 design passes the system bus through both of the two FLEX 10K devices. The system control signals which are now generated by the first CPLD can therefore be altered to suite the timing requirements of the supported components. Due to the fact that both CPLDs (FLEX 10K10s) have 20% spare capacity any future timing modifications can easily be implemented on these devices without the immediate need to migrate to a device with higher densities.

4.2.8 *Electromagnetic Interference Reduction Techniques*

The phase2 design greatly reduces the presence of EMI since the component count is considerably reduced. The technology utilized in the construction of the FLEX 10K CPLDs, also guarantee a large improvement in the internal to external shielding of EMI. The new multi-layer CPLDs also allow the internal busses of the CPLD to operate at very high frequencies without passing unwanted interference to the external circuitry.
The EMI reduction techniques implemented in the phase1 design have also been included in the phase 2 design. The EDA design package , Protel, offered options for reduction in capacitive coupling and cross-talk during PCB routing. All of the interference reduction capabilities offered by Protel were used in the design of the phase2 board.

4.2.9 *Power Analysis and Optimization*

The peak and average power consumption of a circuit are tied to the reliability and integrity of the circuit. Some of the phenomenons effecting the reliability are: electromigration, high resistive volt drops, increased noise margins, increased clock skews and ground bounce. Reducing the circuits peak and average power consumption, has the beneficial side-effects of improving the circuit's reliability. The use of the idle and power-down modes supported by the 80C196KC greatly reduce the power consumption in the design.

The design methods affecting the power optimization of the circuit can be considered at three differe nt stages. The behavioral (functional), structural and physical. The behavioral view specifies the functionality of the design. The structural view represents the circuit as an interconnection of elements or building blocks. The physical view represents the circuit as a set of geometric entities placed an a chip or board. Synthesis tools can be used today to automatically convert or refine a design from a higher level of abstraction to a lower level of abstraction. These tools may refine the design for the reduction of power consumption at the various stages of development. The stages at which this may occur are; the behavioral description (VHDL) stage, the functional RTL stage, the structural RTL stage and the gate-level netlist stage. The benefits of high-level power analysis and optimization using the available tools, are presented in
Figure 4.3.

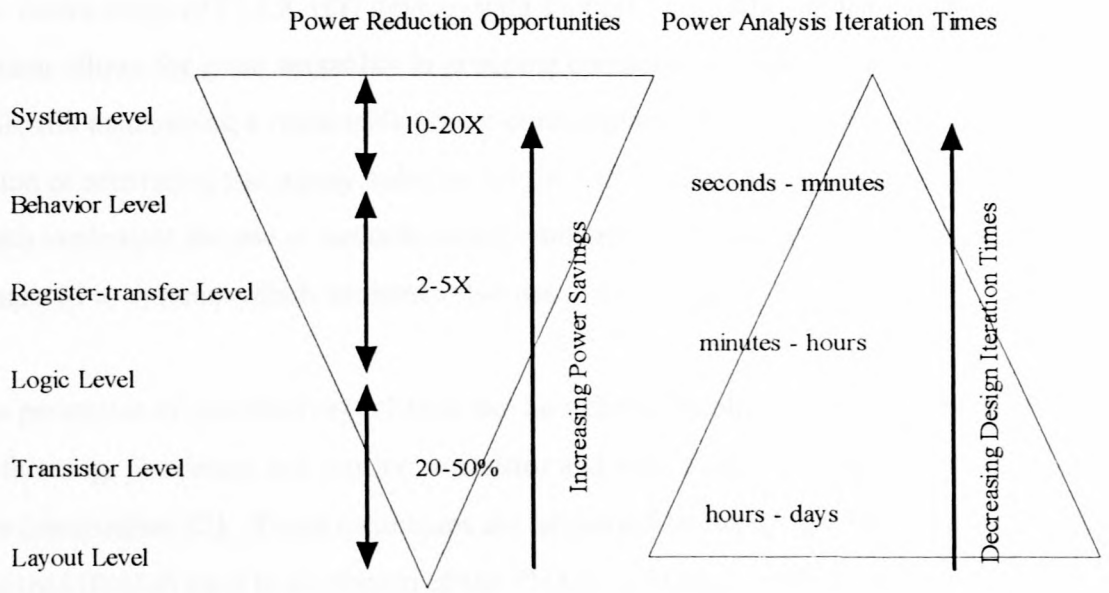


FIGURE 4.3

The parameters to be considered when optimizing the power consumption of a circuit are, the supply voltage, the clock frequency, the switching activity per clock cycle at various signals in the circuit and the parasitic capacitances. In order to reduce the power consumption, one or more of these parameters need to be reduced. If the supply voltage is reduced however, the delay of the circuit elements increase according to the equation.

$$delay = k\frac{Vdd}{(Vdd - V_T)^2}$$

where k is a constant [2].

V_T is the threshold voltage of the transistors

V_{dd} is the supply voltage

The Altera range of FLEX 10K devices with support for interfacing with both 3,3 V and 5 Volts systems allows for great versatility in grouping components requiring different supply voltages, while still maintaining a reasonable power consumption. Altera also allows the designer the option of separating the supply voltages for the I/O circuitry and the chip's core. The system which implement the use of multiple supply voltages have been shown to reduce power as compared to systems which implement the use of single supply voltages [2].

The parameter of switched capacitance can be reduced by physical techniques such as partitioning, placement and routing, transistor and wire sizing, transistor re-ordering and clock tree construction [2]. These techniques are supported by many new EDA tools available today. The tool (Protel) used in the design of the PHASE 2 PCBs provided support for this feature.

The reduction in clock frequency will result in power reductions, but it will not reduce the amount of energy required to perform a task, which is not always beneficial. The idle and power-down modes of the 80C196KC shut down the internal or external clocks in order to reduce power consumption is not necessary. The design incorporates power reduction methods at both board level and chip level using the tools available.

4.2.10 *Cost Effective Design Methodology*

The inclusion of CPLDs into the new design provides the following cost effective advantages :

- Improved functional complexity at no extra cost
- Option to increase the chip density without the additional cost and time for the development of new supporting hardware. This is accomplished by using the chip migration feature.
- Immediate implementation of new designs and upgrades through the reconfigurable nature of the CPLD.
- Internal diagnostics via either pre-configured code or via the download cable.

4.3 The CPLD Implementation into PHASE 2 Design

4.3.1 *Design Implementation using the Hierarchal Interfaces*

The design of the PHASE 2 system has two CPLDs incorporating all of the three hierarchal design levels supported by Altera. The three design levels supported are, the graphical user interface, the text definition files and the symbol files.

4.3.1.1 *The Graphical User Interface*

The entire CPLD configurations are linked through their graphical user interfaces and then compiled. Schematic diagrams of these designs are presented in appendix N.

4.3.1.2 *Text Definition files*

These files were used to include the VHDL and AHDL source code for the following :

- the control logic implemented through symbols GAL1 and GAL2.
- the state machines used for timing alterations, included in symbols GAL1 and GAL2
- the watchdog timer linked to the external reset of the system

Listings of the source code for each of these files is presented in appendix O.

4.3.1.3 *Symbol files*

Altera provides megafunctions which are available for inclusion into the configuration of the CPLD. These megafunctions are generally provided in their symbol form, offering limited access to the VHDL or AHDL code. There are however conditions under which this code can become available to the MAXPLUS II user.

Megafunctions provided by Altera which were used in this project are listed below :

- 1 x a16450 uart (a16450 - MegaCore_ Function))
- 8 x 16 bit counters (LPM COUNTER)
- 6 x 74373 (D LATCHES)
- 1 x 16 bit multiplexers (LPM MUX)

The text definition files containing the control logics, watchdog timers and state machines were also used to generate symbols allowing them to be incorporated into the graphical user interface (GUI). These symbol files are presented in appendix P.

4.4 **Design Methodology using the EPF10K10TC(144)**

After an investigation, it was decided to route the 16-bit busses through both of the two CPLDs. This would allow the bus multiplexing to be performed inside the CPLD. In doing this, any signal alterations, either timing or logic could then be carried out inside the CPLD itself. This provided

the signal versatility required for the design.

A wait-state generator (state machine) , is implemented inside the CPLD in order to adjust the microcontroller timing to suite the components being addressed.

Due to the improved support of LPM_Megafunctions by Altera, it was decided that the following Megafunctions be implemented in the phase 2 design:

- the a16450 uart
- the LPM_COUNTER (a 16-bit counter)
- the LPM_MUX (a 16-bit multiplexer)

These megafunctions are included in the design in symbol form.

The gate count requirement of the two designs led to the selection of one FLEX 10K10 and one FLEX 10K20 device. Since the FLEX 10K range is in support of migration, it is possible to increase the combined densities from 20 000 gates to 100 000 gates. This versatility provides gates sufficient for most possible upgrades, avoiding immediate product obsolescence.

An internal watchdog timer has also been included into the CPLDs. This was done so that the CPLDs could be reconfigured automatically in the event of a fault.
When routing the FLEX 10K devices, multiple ground pins were used to minimize the ground bounce by keeping the lines short and using the internal ground plane of the device.
Critical signals were routed to pins near the groung pins in order to minimize the cross-talk.

The bus signal pins are spread across the device so that simultaneous signal switching on certain pins would not all share the same ground pins. Because of the fact that the use of ic sockets would have resulted in an increase in unwanted disturbances due to an increase in the device lead length, the components were soldered directly onto the printed circuit board.

The MAXPLUS II software tool provides the designer with an option of adjusting the slew rate of the FLEX 10K CPLD. By slowing down the slew rate of the CPLD, it is possible to reduce the ringing in the CPLD. This option does however result in a sacrifice in speed.

It must be noted that this option is not supported by all PLDs, and therefore influenced the selection of the chosen device used in this project.

4.5 Bi-directional Tri-state Bus Implementation into the EPF10K10TC(144)

Due to the fact that Altera devices do not have internal tri-state buses, logic inside a device cannot communicate directly via bidirectional buses. MAXPLUS II can emulate tri-state buses by using multiplexers and by routing the bidirectional line outside of the device and then back in through another pin.
In this project, the tri-state buses are used to multiplex signals internally.

MAXPLUS II converts the logic to a combinational multiplexer. Figure 4.4 represents the tri-state buses for bidirectional communication. Device 1 communicates with device 2 via bidirectional lines.

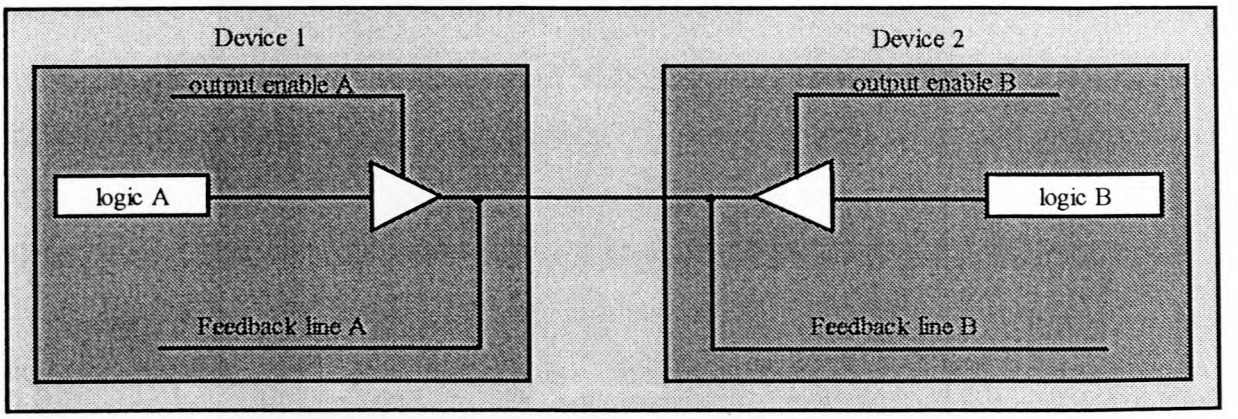


FIGURE 4.4

This communication can be performed inside an Altera PLD since there is a tri-state buffer available at each I/O pin.

If the communication occurs via bidirectional lines inside the device, it cannot be implemented as indicated in figure 4.5.

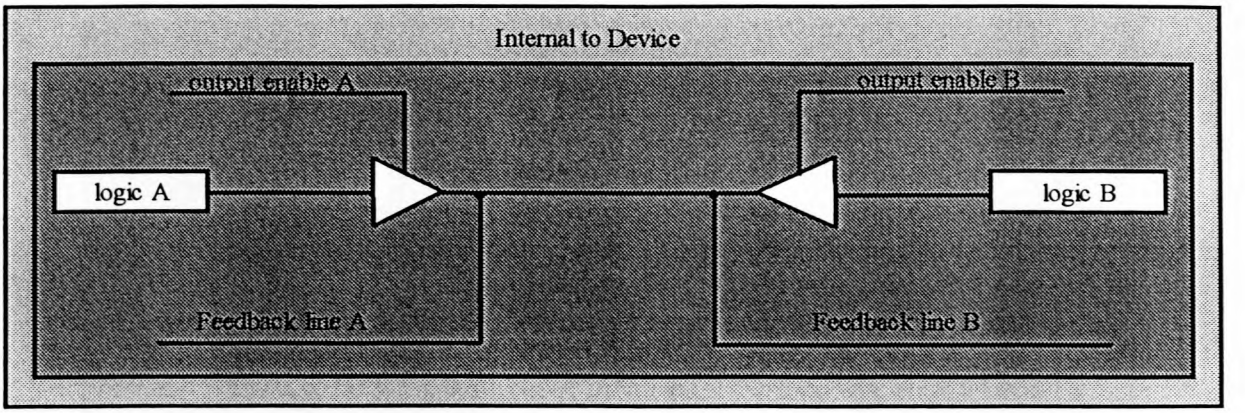


FIGURE 4.5

In order to overcome this limitation, one must route the bidirectional lines outside of the device. This is presented in figure 4.6.

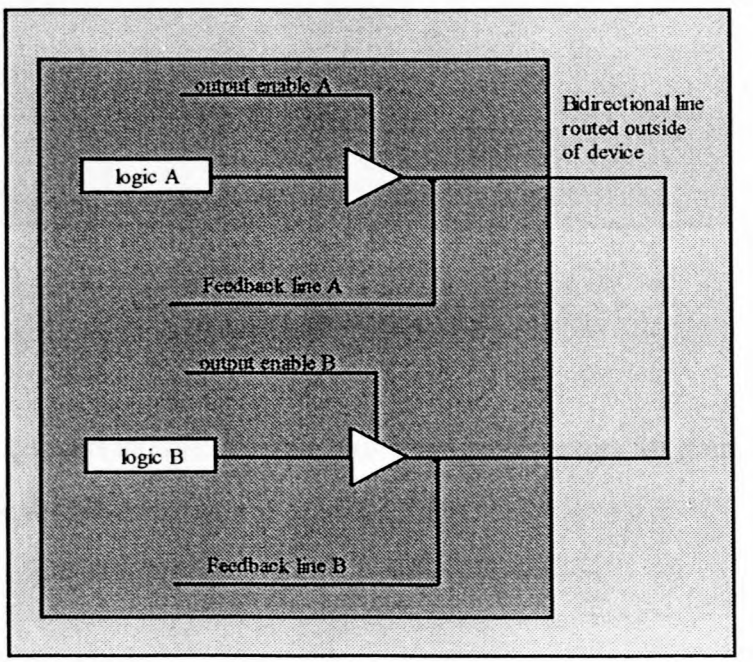


FIGURE 4.6

This method makes use of tri-states present at the I/O pins. This Phase 2 design does however not allow for the bus signals to use the I/O pins as an intermediate register for multiplexing, due to the fact that there would then be a shortage of I/O pins.

Another method is to convert the tri-state buses into a multiplexer as shown in figure 4.7.

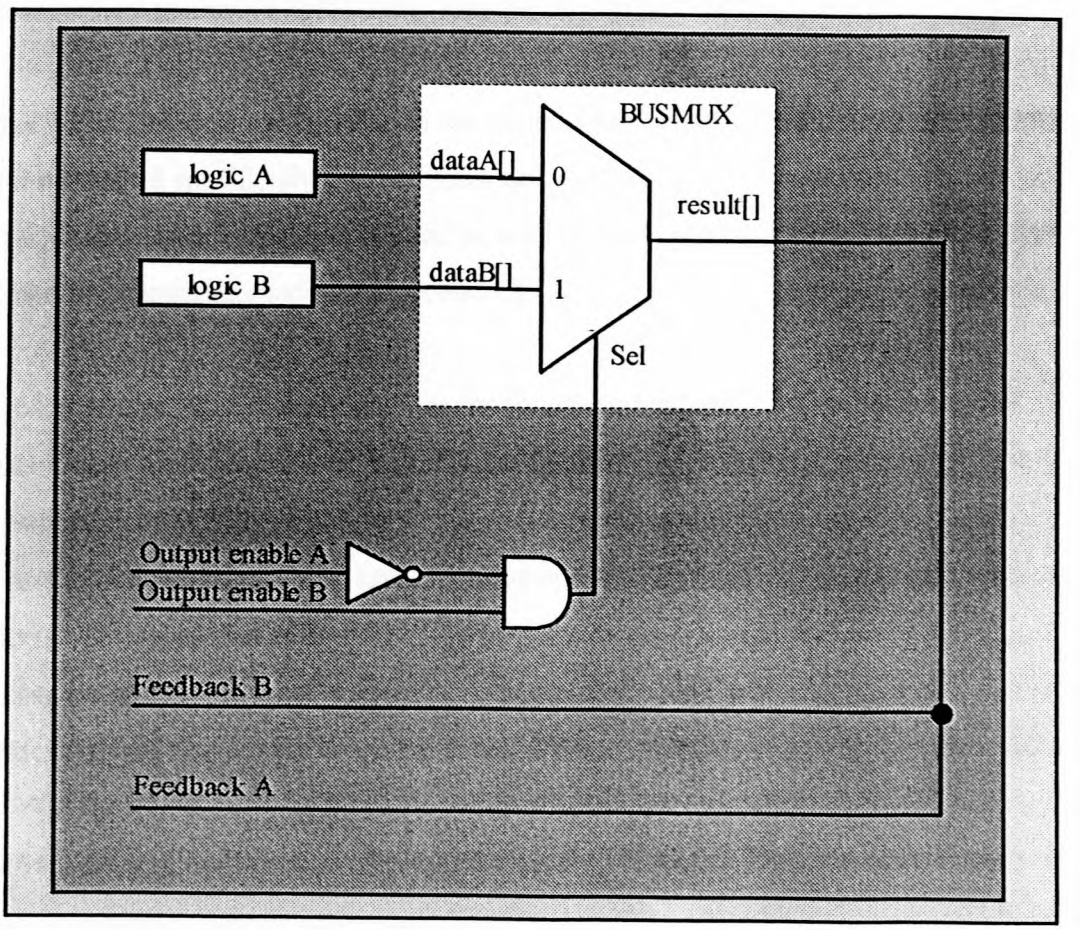


FIGURE 4.7

MAXPLUS II can convert internal tri-state buses into multiplexers for graphic editor designs as used in this phase 2 design.

4.6 The a16450 MegaCore Function

This device is implemented in one of the two CPLDs in order to reduce the physical component count on the main board. It replaces the Intel 82510 uart set on the PCB.
The choice of function was due to the fact that the 82510 uart has it's first set of addressable registers identical to the NS16450 uart. This meant that the existing control circuitry could still be used.
The FLEX 10K device is designed with full support for MegaCore functions, with the a16450 function optimized specifically for the FLEX architecture.
The FLEX10k10 device however proved to insufficient in terms of gate density and the FLEX 10k20 had to be implemented as the second CPLD.

The a16450 MegaCore function includes the following features:
- programmable word length, stop bits and parity bits
- programmable baud rate generator
- supports both synchronous and asynchronous operation
- supports full duplex operation
- supports prioritized interrupt control
- provides internal diagnostic/ loopback capabilities

The a16450 MegaCore function uses approximately 372 FLEX logic elements (Les)

Due to the fact that two different baud rates are required from the a16450 for communication with the Dallas touch memory, a uart with a fully programmable baud rate generator was chosen.

The a16450 was simulated and found to be operating correctly. The MegaCore function could however not be implemented immediately into the project, since a license for the function would first have to be purchased from the AMPP suppliers.

4.7 Synthesis ans Routing of the EPF10K10/20TC(144) Internal Designs

The compiler uses a default logic synthesis style for the entire project. Altera provides Normal, Fast and WYSIWYG logic synthesis styles which specify a setting for every available individual logic option. Altera does however not recommend editing the logic option settings in these styles. Because the MegaCore functions used for the FLEX 10K designs contain cascade and carry buffers, a style other than Normal is used as the default logic synthesis style.

Altera's logic synthesizer provides two basic types of synthesis ;
the multi-level logic synthesis and the standard synthesis.

In this project the multi-level synthesis option is used.

The results are reflected in the synthesis and routing outputs presented in appendix Q.

4.8 Simulation of the EPF10K10TC(144) and EPF10K20TC(144)'s internal design

The internal operation of the both of the CPLDs were simulated and are operating correctly.

4.9 System analysis of the EPF10K10TC(144) and EPF10K20TC(144) CPLDs

The timing analysis is presented in appendix S.

4.10 Cascaded Phase 2 CPLD configuration on power up

It was found that the 80C196 microcontroller created bus interference while the CPLDs were in their configuration cycle. The entire system excluding the CPLDs were issued a global reset after the CPLDs had been initialized, but it was found that since the 196 was operational during part of the CPLDs cinfiguration process, interference still occurred and the ONCE command was then used to electrically isolate the 80C196 microcontroller from the rest of the board until the CPLD had completed it's initialization cycle. This proved successful.

4.11 Manufacture and Implementation of the Phase 2 System

4.11.1 Printed Circuit Board Design

The schematics were drawn by myself in the Protel EDA environment. The schematics are presented in appendix U.

The netlist files were generated in Protel and transferred to the Protel router. The frequencies and voltages were specified for the routing process and the Software routed to minimize the crosstalk and voltage interferences. The routing guidelines mentioned earlier in the thesis were implemented in order to produce preferred results.

The entire design previously required three PCBs, whereas with the new CPLD technology, the design has been reduced to a single board with a much reduced component count.

The printed circuit board containing all of the required hardware is presented in appendix U

4.11.2 System installation

The Phase 2 system has replaced the previous Phase 1 system and was found to be more reliable and easier to install. With fewer components to test and fewer boards to populate, the installation has been much improved.

Chapter 5

Conclusions, Recommendations and Future Work

5.1 **Conclusions**

The Phase1 system initially seemed easier to implement, but in hindsight it was realized that this was due to the relatively new and unknown technology used in the Phase 2 system.
The phase 2 design which includes the CPLDs could be simulated and tested in much less time than the Phase 1 design. It is thus fair to say, that a big advantage of using the newer CPLD technology, is a greatly reduced time-to-market of the product. This is largely due to the drive toward rapid prototyping in order to avoid product obsolescence.
The Phase 2 system proved to be the more electrically stable system. This can also be attributed to the alterations in the design process based on the recommendations following the problems encountered in the phase 1 system.
The reduction in electrical noise in the later design can also be attributed to the construction of the CPLD as mentioned in the thesis.
The MegaCore function offers a means of testing the new uart before a purchase is made. This allows the designer the option of testing a far greater range of products before finally deciding on the most suitable one for the application.
The CPLDs support of JTAG and BST technology now also allows the designer to perform continuity tests on the PCB without wasting time with bare-board testing.
One important consideration still to be made in designing with CPLDs, is that the designer must over-design with the support hardware in order to utilize the migration features of the CPLDs.
It also important to note that newer CPLDs are becoming faster and at the same time minimizing

the interferences.

5.2 Recommendations

The comparison between technologies used in this thesis leads to the recommendation that were feasible, all of the newer designs make use of the newer CPLD technology. It will in future become increasingly difficult to compete in the electronic design market unless the designer strives to avoid short term product obsolescence. One of the ways to assist in this regard is to use a system which supports density migration and fast upgrades (e.g. by using the in-circuit programmability feature of Altera's CPLDs)
With the increase in densities of the newer CPLDs, it is inevitable that the future holds a complex system on a chip immediately adaptable to every designer's requirements.

Altera does however need to provide a greater support for the development of an internal tri-state bus, eliminating the need for the use of multiplexers.

5.3 Future work

There remains a need for greater control of the driving capabilities on the I/O pins of the device itself. I believe this is one of the areas where some of the large FPGA/CPLD manufacturers need to do some development.

The AMPP program also needs to grow in order to provide a greater range of products required for the Mega function usage to develop.

Bibliography

[1] Stefan Sjoholm and Lennart Lindh, *VHDL for Designers*. Prentice Hall, Europe, 1997.

[2] Anand Raghunathan, Niraj K. Jha and Sujit Dey,

[3] Zoran Salcic and Asim Smailagic, *Digital Systems Design and Prototyping using Field Programmable Logic*. Kluwer Academic Publishers, Massachusetts, USA, 1998.

[4] Ramesh S. Goankar, *Microprocessor Architecture, Programming and Applications*. 2nd Edition. Merril Publishing Company, Columbus, Ohio, 1989.

[5] Intel Corporation, *8XC196KC/8XC196KD User's Manual*, USA, 1992.

[6] Radiometrix Limited, "*Low Power UHF Data Transceiver Module*", in BIM-UHF Data Sheet, Issue 3, 21st October 1997.

[7] Altera Corporation, "*Introduction to Programmable Logic and ASICs*", in Altera documentation, January 1998, Version 5.

[8] Steve Money, *Newnes Microprocessor Pocket Book*. Heinemann Newnes Publishing, Oxford, 1989.

[9] Intel Corporation, *EV80C196KD Evaluation Board User's Manual*, USA, Release 1, 8 May 1992.

[10] Altera Corporation, "*FLEX 10K Embedded Programmable Logic Family*", in Altera documentation, October 1998, Version 3.13.

[11] C.H. Jordaan, "*the Implementation of a Microprocessor in a FPGA*", in Thesis for Masters Degree in Electronic Engineering of University of Stellenbosch, February 1998.

[12] G.C. Loveday, *Designing Electronic Hardware*, Longman Scientific and Technical, UK, 1992.

[13] Thomas C. Hayes and Paul Horowitz, *The Art of Electronics*, Cambridge University Press, USA, 1989.

[14] Dan Garish and Tsvika Kurts, *High Performance driver for 82510*, Application Note 310, Intel Corporation, June 1987.

[15] Faisal Imdad-Haque, *Designing with the 82510 Asynchronous Serial Controller*, Application Note 401, Intel Corporation, December 1987.

[16] John Adams, *Techniques for accurate time measurement with the Dallas Semiconductor DS1994 Time-in-a-Can*, Application Note 60, Dallas Semiconductor, March 1993.

[17] Dallas Semiconductor, *Understanding and Using Cyclic Redundancy Checks with Dallas Semiconductor Touch Memory Products*, Application Note 27, Dallas Semiconductor, October 1995.

[18] Dallas Semiconductor, *64K NV SRAM with Phantom Clock*, Dallas Semiconductor Data Sheet on DS1243Y, December 1995.

[19] Tom Williams, *Designing Microcontroller Systems for Electrically Noisy Environments*, Application Note 125, Intel Corporation, December 1993.

[20] Altera, *1995 Databook*, San Jose, CA, 1995.

[21] Intel, *16-Bit Embedded Controller Handbook*, Santa Clara, 1989.

[22] Intel, *8-Bit Embedded Controller Handbook*, Santa Clara, 1990.

[23] Intel, *Embedded Microcontrollers Databook*, USA, January 1998.

[24] Neil Slay, *Safety-Critical Computer Systems*, Addison Wesley Longman Inc, New York, 1996.

[25] Von L. Burton, *The Programmable Logic Device handbook*, TAB Book Inc, USA, 1990.

[26] Geoff Bostock, *FPGAs and Programmable LSI : A Designer's Handbook*, Butterworth-Heinemann, London, 1996.

[27] R.C. Seals and G.F. Whapshott, *Programmable Logic : PLDs and FPGAs*, MacMillan Press Ltd, Great Britain, 1997.

[28] Parag K. Lara, *Digital System Design using Programmable Logic Devices*, Prentice Hall, Eaglewood Cliffs, New Jersey, 1990.

[29] Kenneth J. Hintz and Daniel Tabak, *Microcontrollers : Architecture, Implementation and Programming*, McGraw-Hill Inc, USA, 1992.

[30] John W. Carter, *Digital Designing with Programmable Logic Devices*, Prentice-Hall, Inc, New Jersey, 1997.

[31] Geoff Bostock, *Programmable Logic Handbook*, 2nd Edition, Newnes, Great Britain, 1993.

[32] Altera Corporation, "*IEEE 1149.1 (JTAG) Boundary-Scan Testing in Altera Devices*", Version 4.02, in Altera Application Note 39, November 1998.

[33] Altera Corporation, "*FLEX 10K Device Family*", in Altera Documentation, March 1999.

[34] Nathan Gurewich and Ori Gurewich, *Teach yourself Visual Basic 4 in 21 Days*, 3rd Edition, Sams Publishing, USA, 1995.

[35] Boston Systems Office/Tasking, *Software Development Under Control*, Tasking Software BV, The Netherlands, 1994.

[36] Siemens, "*Plastic Fiber Components (PFC): A Cost-Effective Sollution for Optical Signal Transmission*", Application Note 40, 1997.

[37] Siemens, "*Plastic Fiber Optic Phototransistor Detector*", Siemens Data Sheet.

[38] Siemens, "*Plastic Fiber Optic Phototransistor Diode*", Siemens Data Sheet.

[39] Intel, '*8251A Programmable Communication Interface*", Santa Clara, November 1986.

[40] Altera Corporation, "*Byte Blaster Parallel Port Download Cable*", Version 2.01, in Altera Data Sheet, February 1998.

[41] Altera Corporation, "*Configuration EPROMS for FLEX Devices*", Version 9, in Altera Data Sheet, October 1998.

[42] Motorola Semiconductor Technical Data, "*12-Stage Binary Ripple Counter-High-Performance Silicon-Gate CMOS*", Motorola Literature Distribution, Arizona, Revision 1, 1995.

[43] Altera, "*Reflow Soldering Guidelines for Surface-Mount Devices*", Version 3, in Altera Application Note 81, January 1999.

[44] Altera, "*Implementing Tri-States Buses in Altera Devices*", Altera Application Note, webmaster@altera,com, http://www.altera.com/html/atlas/examples/ged/tri-state.html, 9 February 1998.

[45] Altera, "*AHDL: Tri-State Buses Connected to a Bidirectional Bus*", Altera Application Note, webmaster@altera.com, http://www.altera.com/html/atlas/examples/ahdl/a-tri-bb.html, 9 February 1998.

[46] Altera, "*Synthesizing and Optimizing VHDL and Verilog HDL with Synopsis Software*", Altera Application Note, webmaster@altera.com, http://www.altera.com/html/maxkey/synopsis/compilers/vsynt.htm, 29 January 1998.

[47] Altera, "*Running Synopsis Compilers from the MAX+PLUS II Software*", Altera Application Note, webmaster@altera.com, http://www.altera.com/html/maxkey/synopsis/compilers/syncom.htm, 29 January 1998.

[48] Altera, "*Setting up the MAX+PLUS II/Synopsis Working Environment*", Altera Application Note, webmaster@altera.com, http://www.altera.com/html/maxkey/synopsis/intro/setup.htm, 29 January 1998.

[49] Altera, "*MAX+PLUS II/Synopsis Interface File Organization*", Altera Application Note, webmaster@altera.com, http://www.altera.com/html/maxkey/synopsis/intro/fileorgn.htm, 29 January 1998.

[50] Altera, "*Altera Packaging Solutions: Package Outlines*", Altera Application Note, webmaster@altera.com, http://www.altera.com/html/atlas/packages1.html#code, 26 March 1999.

[51] Altera, "*98251-Programmable Communications Interface*", Version 2, Altera Data Sheet, June 1997.

[52] Altera, "*Operating Requirements for Altera Devices*", Version 8, Altera Data Sheet, January 1998.

[53] Altera, "*Bit Blaster Serial Download Cable*", Version 4.01, Altera Data Sheet, February 1998.

[54] National Semiconductor, "*LM311 Comparator General Description*", National Semiconductor Corporation, Data Sheet, USA, 1995.

[55] Harris Semiconductor, "*CA3140 - 4,5 MHz, BiMOS Operational Amplifier with MOSFET Input/Bipolar Output*", Harris Corporation, Data Sheet, File Number 957.4, 1998.

[56] National Semiconductor, "*CD4093BM Quad 2-Input NAND Schmidt Frigger*", National Semiconductor Corporation, Data Sheet, USA, February 1993.

[57] National Semiconductor, "*CD4001BM Quad 2-Input NOR Buffered B Series Gate*", National Semiconductor Corporation, Data Sheet, USA, March 1998.

[58] Motorola Semiconductor Technical Data, "*MC14C88B-Quad Low Power Line Driver*", Motorola Literature Distribution, Revision 0, Arizona, 1996.

[59] Motorola Semiconductor Technical Data, "*MC1489A - Quad Low Power Line Driver*", Motorola Literature Distribution, Revision 5, Arizona, 1996.

[60] Dallas Semiconductor, "*How to Save Data During a Power Failure without Corrupting it*", Application Note 51, Dallas Semiconductor, April 1998.

[61] Dallas Semiconductor, "*Using the Phantom Real Time Clocks*", Application Note 52, Dallas Semiconductor, February 1994.

[62] James W. McCord, *Developing Windows Applications with Borland C++3.1*, 2nd Edition, SAMS Publishers, USA, 1992.

[63] Dallas Semiconductor, *High-Speed Microcontroller Data Book*, USA, 1995.

[64] Dallas Semiconductor, *DS1996 64k bit Touch Memory*, Dallas Semiconductor Data Sheet, February 1995.

[65] Siemens, *"Fiber Optic Links at Lowest Cost : Plastic Fiber Components"*, Siemens Literature Sheet, http://www.siemens.de/semiconductor/products/37/37741.htm, 1998.

[66] David Hinerman, *"Making Non-volatile Data Reliable"*, Microcom Corporation, Westerville, OH, 1997.

[67] Altera Corporation, *"Using Synopsys FPGA Express Software to Synthesize Designs for MAX+PLUS II Software"*, Altera Technical Brief 42, Version 1, 1998.

[68] Altera Corporation, *"Using Synopsis Compiler & FPGA Compiler to Synthesize Designs for MAX+PLUS II Software"*, Altera Technical Brief 39, Version 1, April 1998.

[69] Altera Corporation, *"Generating Post-Route Files in the MAX+PLUS II Software for Third-Party Verification Tools"*, Altera Technical Brief 49, Version 1.01, October 1998.

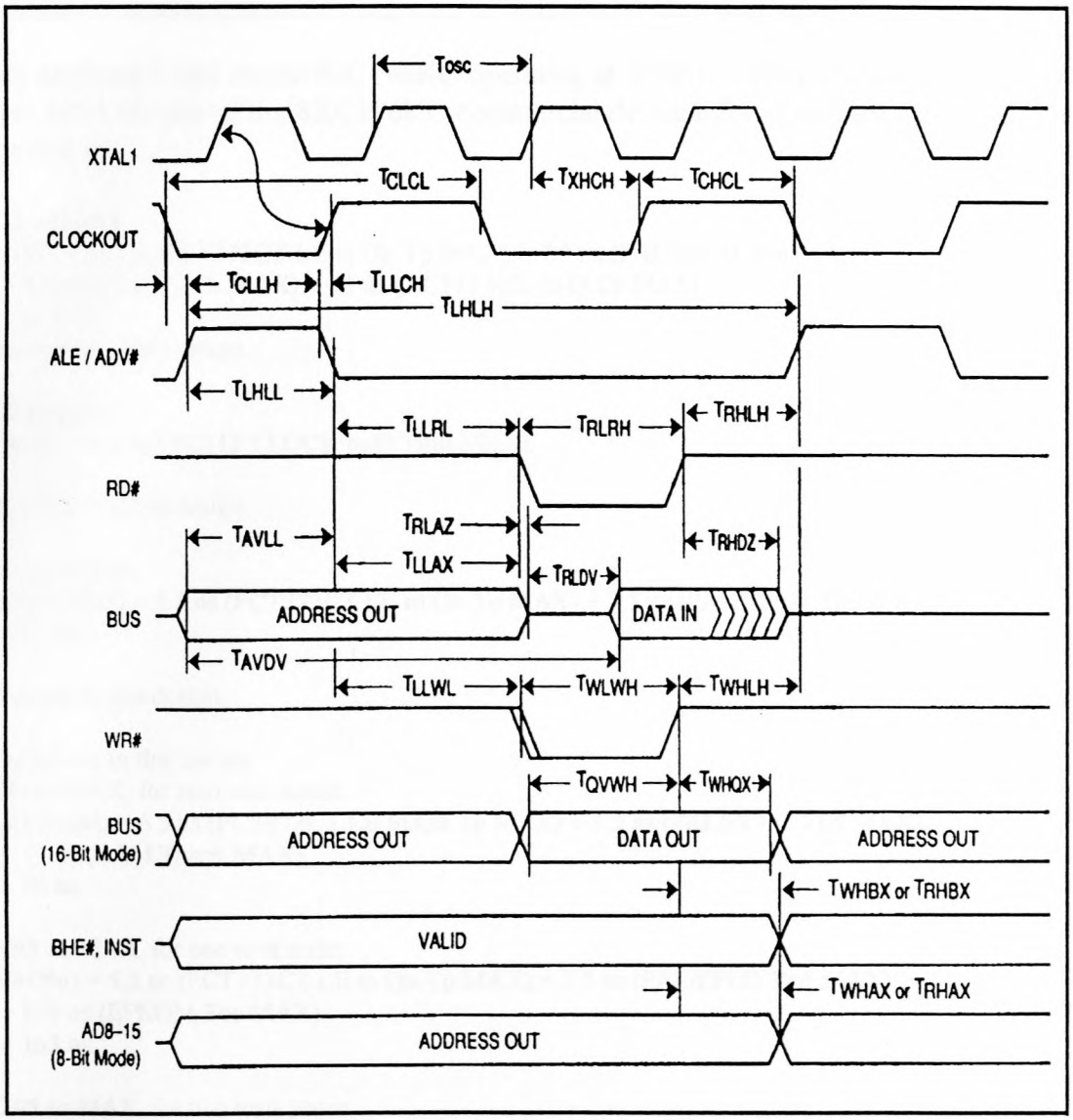
[70] Altera Corporation, *"Passing Hierarchical Timing Constraints from Synopsys Tools to MAX+PLUS II Version 9"*, Altera Technical Brief 48, Version 1, 1998.

[71] Altera Corporation, *"Importing Synthesized Files from EDA Tools into the MAX+PLUS II Software for Place and Route"*, Altera Technical Brief 45, Version 1, 1998.

Appendices

Appendix A-1

System Bus Timing



Appendix A-2

Timing Analysis of the 80C196KC System Board

All values used are based on the 80C196KC operating at 20MHz. They are taken from the November 1994 version of the 8XC196KC commercial device information data sheet. The Intel order number is:

Tavyv = 32 ns MAX
Tavyv(WAIT) = 5.5 ns (FCT373CT LE to Qn Tp MAX) + 7.5 ns (PAL/EPLD Tpd MAX)
+ 7.9 ns (AC08 Tplh MAX) + 11 ns (AC112 RES to Q Tp MAX)
= 31.9 ns.
Tllyv is irrelevant in this design.

Tclyx = 20 ns MAX.
Tclyx (WAIT) = 9.4 ns (AC112 CLOCK to Q Tplh MAX).

Tllyx is irrelevant in this design.

Tavgv = 32 ns MAX.
Tavgv(BUSWIDTH) = 5.5 ns (FCT373CT LE to Qn Tp MAX) + 7.5 ns (PAL/EPLD Tpd MAX)
= 13 ns.

Tllgv irrelevant in this design.

Tclgx is irrelevant in this design.
Tavdv = 95 ns MAX, for zero wait states.
Tavdv(16bit RAM) = 5.5 ns (FCT373CT LE to Qn Tp MAX) + 7.5 ns (PAL/EPLD Tpd MAX)
+ 45 ns (RAM Tavqv MAX)
= 58 ns.

Tavdv = 195 ns MAX, for one wait state.
Tavdv(EPROM) = 5.5 ns (FCT373CT LE to Qn Tp MAX) + 7.5 ns (PAL/EPLD Tpd MAX)
+ 150 ns (EPROM Tce MAX)
= 163 ns.

Tavdv = 295 ns MAX, for two wait states.
Tavdv(UART) = 5.5 ns (FCT373CT LE to Qn Tp MAX) + 7.5 ns (PAL/EPLD Tpd MAX)
+ 190 ns (UART Tavrl MIN + Trldv MAX)
= 203 ns.

Trldv = 28 ns MAX, for zero wait states.
Trldv(16bit RAM) = 20 ns (RAM Tglqv MAX).

Trldv = 78 ns MAX, for one wait state.

Trldv(EPROM) = 60 ns (EPROM Toe MAX).

Trldv = 178 ns MAX, for two wait states.
Trldv(UART) = 173 ns (UART Trldv MAX).

Tcldv is irrelevant in this design.

Trhdz = 50 ns MAX.
Trhdz(16bit RAM) = 25 ns (RAM Tghqz MAX).
Trhdz(EPROM) = 50 ns (EPROM Tdf MAX).
Trhdz(UART) = 40 ns (UART Trhdz MAX).

Trxdx = 0 ns MIN.
Trxdx(16bit RAM) = 0 ns (RAM Tghqz MIN).
Trxdx(EPROM) = 0 ns (EPROM Toh MIN).
Trxdx(UART) is not specified.

Txhch is irrelevant in this design.

Tclcl = 100 ns.
Tclcl(WAIT) = 9 ns (PAL/EPLD Tp MIN).
= 9 ns (AC112 1/Fmax MIN).

Tchcl = 40 ns MIN.
Tchcl(WAIT) = 6 ns (PAL/EPLD Tco2 MAX) + 4 ns (AC112 Ts MIN)
= 10 ns.
or = 6 ns (PAL/EPLD Tco2 MAX)
+ 7.9 ns (AC08 Tplh MAX) + 3 ns (AC112 Trec MIN)
= 16.9 ns.

Tcllh is irrelevant in this design.

Tllch is irrelevant in this design.

Tlhlh is irrelevant in this design.

Tlhll = 40 ns MIN.
Tlhll(A0-A15) = 5 ns (FCT373CT Tw MIN).

Tavll = 35 ns MIN.
Tavll(A0-A15) = 2 ns (FCT373CT Ts MIN).
Tavll(WAIT) = 5.5 ns (FCT373CT LE to Qn Tp MAX) + 7.5 ns (PAL/EPLD Tpd MAX)
+ 8 ns (AC00 Tphl MIN) + 5 ns (AC112 Tw MIN)
= 26 ns.
Tavll(BHE#) = 11 ns (AC14 Tplh MAX) + 5 ns (AC112 Tw MIN)
= 16 ns.

Tllax = 10 ns MIN.
Tllax(A0-A15) = 1.5 ns (FCT373CT Th MIN).
Tllax(BHE#) = 0 ns (AC112 Th MIN).
Tllrl = 20 ns MIN.
Tllrl(UART) = 7 ns (UART Tavrl MIN).

Trlcl is irrelevant in this design.

Trlrh = 245 ns MIN, for two wait states.
Trlrh(UART) = 173 ns (UART Trlrh MIN).

Trhlh = 50 ns MIN.
Trhlh(STALE) = 9 ns (AC08 Tplh MAX) + 3 ns (AC112 Trec MIN)
= 12 ns.

Tllwl = 40 ns MIN.
Tllwl(UART) = 7 ns (UART Tavwl MIN).

Tclwl is irrelevant in this design.

Tqvwh = 27 ns MIN, for zero wait states.
Tqvwh(16bit RAM) = 25 ns (RAM Tdvwh MIN).

Tqvwh = 227 ns MIN, for two wait states.
Tqvwh(UART) = 90 ns (UART Tdvwh MIN).

Tchwh is irrelevant in this design.

Twlwh = 30 ns MIN, for zero wait states.
Twlwh(16bit RAM) = 25 ns (RAM Twlwh MIN).

Twlwh = 230 ns MIN, for two wait states.
Twlwh(UART) = 123.5 ns (UART Twlwh MIN).

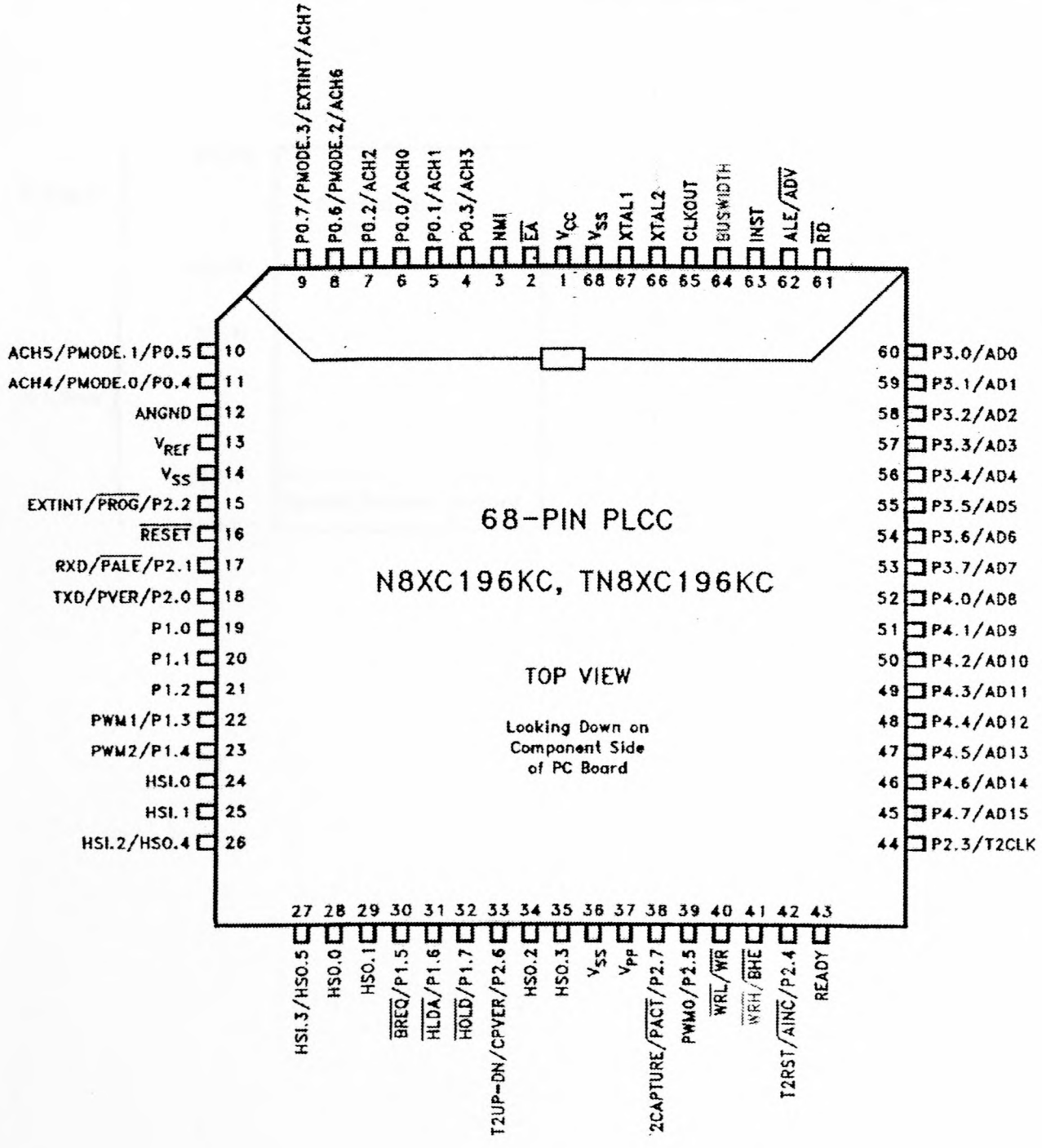
Twhqx = 25 ns MIN.
Twhqx(16bit RAM) = 9 ns (74AC32 Tplh MAX) + 0 ns (RAM Twhdx MIN)
= 9 ns.
Twhqx(U14) = 0 ns (RAM Twhdx MIN).
Twhqx(UART) = 12 ns (UART Twhdx MIN).

Twhlh = 40 ns MIN.
Twhlh(16bit RAM) = 9 ns (74AC32 Tplh MAX) + 5 ns (RAM Twhax MIN)
= 14 ns.
Twhlh(UART) = 0 ns (UART Twhax MIN).
Twhlh(STALE) = 9 ns (AC08 Tplh MAX) + 3 ns (AC112 Trec MIN)
= 12 ns.

Twhbx is irrelevant in this design.

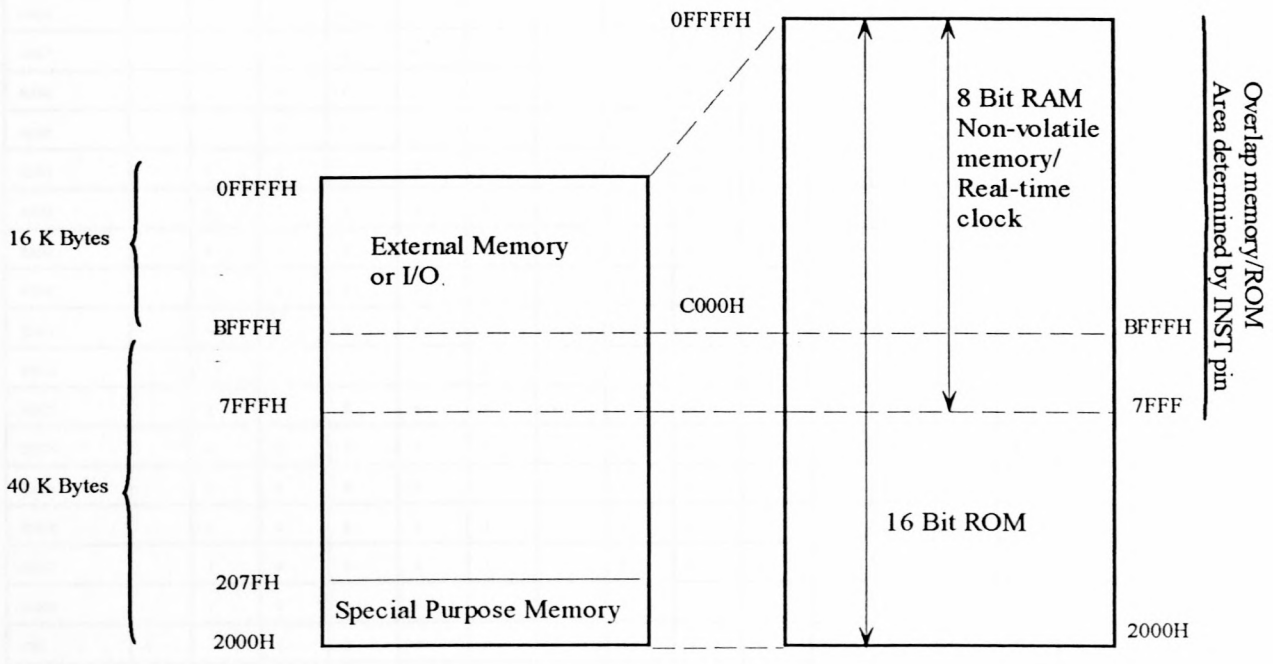
Appendix B

80C196KC 68 Pin PLCC Package



Appendix C-1

Memory Map for Peripheral Outputs



Appendix C-2

Memory Map of PHASE 1 and PHASE 2 Control Signals

	RD	WR	A15	A14	A13	A12	A11	A10	A9	A8	A7	A5	A4	A3	A2	A1	Address
RES8		1	0	0	0	1	1	1	0	0	0	0	1	0	1	1	0X1C16
RES7		1	0	0	0	1	1	1	0	0	0	0	1	0	1	0	0X1C14
RES6		1	0	0	0	1	1	1	0	0	0	0	1	0	0	1	0X1C12
RES5		1	0	0	0	1	1	1	0	0	0	0	1	0	0	0	0X1C10
RES4		1	0	0	0	1	1	1	0	0	0	0	0	1	1	1	0X1C0E
RES3		1	0	0	0	1	1	1	0	0	0	0	0	1	1	0	0X1C0C
RES2		1	0	0	0	1	1	1	0	0	0	0	0	1	0	1	0X1C0A
RES1		1	0	0	0	1	1	1	0	0	0	0	0	1	0	0	0X1C08
ENC1		1	0	0	0	1	1	1	0	0	0	0	1	1	0	1	0X1C1A
ENC2		1	0	0	0	1	1	1	0	0	0	0	1	1	1	0	0X1C1C
ENC3		1	0	0	0	1	1	1	0	0	0	0	1	1	1	1	0X1C1E
ENC4		1	0	0	0	1	1	1	0	0	0	1	0	0	0	0	0X1C20
ENC5		1	0	0	0	1	1	1	0	0	0	1	0	0	0	1	0X1C22
ENC6		1	0	0	0	1	1	1	0	0	0	1	0	0	1	0	0X1C24
ENC7		1	0	0	0	1	1	1	0	0	0	1	0	0	1	1	0X1C26
ENC8		1	0	0	0	1	1	1	0	0	0	1	0	1	0	0	0X1C28
CE1	1		0	0	0	1	1	1	0	0	0	0	0	0	1	1	0X1C06
CE2	1		0	0	0	1	1	1	0	0	0	0	0	0	1	0	0X1C04
CE3	1		0	0	0	1	1	1	0	0	0	0	0	0	0	1	0X1C02
CE4	1		0	0	0	1	1	1	0	0	0	0	0	0	0	0	0X1C00
CE5	1		0	0	0	1	1	1	0	0	0	1	0	1	0	1	0X1C2A
CE6	1		0	0	0	1	1	1	0	0	0	1	0	1	1	0	0X1C2C
CE7	1		0	0	0	1	1	1	0	0	0	1	0	1	1	1	0X1C2E
CE8	1		0	0	0	1	1	1	0	0	0	1	1	0	0	0	0X1C30
BIT16LED		1	0	0	0	1	1	1	0	0	0	0	0	0	1	0	0X1C04
CELED		1	0	0	0	1	1	1	0	0	0	0	0	0	1	1	0X1C06
CSRELAY		1	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0X1C00
EXLAMP		1	0	0	0	1	1	1	0	0	0	0	1	1	0	0	0X1C18
EXFAN		1	0	0	0	1	1	1	0	0	0	1	1	0	0	1	0X1C32
NOZZLE		1	0	0	0	1	1	1	0	0	0	1	1	0	1	0	0X1C34
SBUZZER		1	0	0	0	1	1	1	0	0	0	1	1	0	1	1	0X1C36
SWD1		1	0	0	0	1	1	1	0	0	0	1	1	1	0	0	0X1C38
LEDBANK		1	0	0	0	1	1	1	0	0	0	1	1	1	0	1	0X1C3A
PORT2_RI		1	0	0	0	1	1	1	0	0	0	1	1	1	1	0	0X1C3C
TESTCPLD		1	0	0	0	1	1	1	0	0	0	1	1	1	1	1	0X1C3E

Appendix C-3

Memory Map of PHASE 1 and PHASE 2 Control Signals for GAL 1

Signal	Description	Address	Wait States	Active Low/High
Ce0	Chip select for 16-bit eproms	2000H-BFFFH	1	L
Ce1	Chip select for 16-bit status ram	0000H-0FFFH 1000H-11FFH	0	L
Ce2	Chip select for 8-bit real-time clock/nv ram	C000H-FFFFH	2	L
Cs510	Chip select for 8251A usart	IE00H-IEFFH	2	L
Buswidth	8/16 bit buswidth selection	IE00H-IEFFH & C000H-FFFH	0	H-167 L-8
inst	Memory overlap signal, distinguishes between instruction and data	Only implemented in phase 2 design	0	L

Appendix D

Intel 82510 UART Register Map

- **Refer to bibliography for reference**

Appendix E

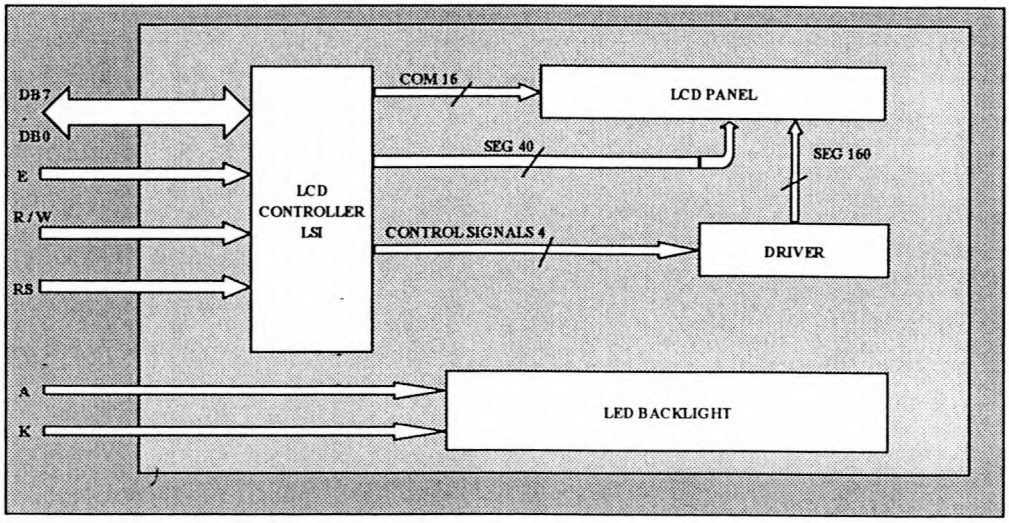
Phantom Clock Control Sheet

Bcd/hex	Bcd/hex	7	6	5	4	3	2	1	0	O/R		
		0.1 sec				0.01 sec						0.1 sec
0	0	0	0	0	0	0	0	0	0	#	A	
9	9	1	0	0	1	1	0	0	1	A	#	
		10 sec				SECONDS						SECONDS
0	0	0	0	0	0	0	0	0	0			
5	9	0	1	0	1	1	0	0	1			
		10 min				MINUTES						MINUTES
0	0	0	0	0	0	0	0	0	0			
5	9	0	1	0	1	1	0	0	1			
		12		A/P	HR	HOUR						HOURS 12 H MODE
0	1	1	0	0	0	0	0	0	1			1 AM
1	2	1	0	0	0	1	0	0	1			9 AM
		1	0	0	1	0	0	0	0			10 AM
		1	0	0	1	0	0	1	0			12 AM
		1	0	1	0	0	0	0	1			1 PM
		1	0	1	0	1	0	0	1			9 PM
		1	0	1	1	0	0	0	0			10 PM
		1	0	1	1	0	0	1	0			12 PM

		24		20								24 H MODE
0	0	0	0	0	0	0	0	0	0			1 AM - 01:00
2	4	0	0	0	0	1	0	0	1			9 AM - 09:00
		0	0	0	1	0	0	0	0			10 AM - 10:00
		0	0	0	1	0	0	1	0			12 AM - 12:00
		0	0	0	1	0	0	1	1			1 PM - 13:00
		0	0	1	0	0	0	0	0			8 PM - 20:00
		0	0	1	0	0	0	1	1			11 PM - 23:00
		0	0	0	0	0	0	0	0			12 PM - 00:00
				OSC	RST		DAY					DAY
0	1	0	0	0	1	0	0	0	1			SUNDAY
0	7	0	0	0	1	0	1	1	1			SATURDAY
#OSC AND # RST ARE ACTIVE LOW												
				10 Date		DATE						DATE
0	1	0	0	0	0	0	0	0	1			01
3	1	0	0	1	1	0	0	0	1			31
					10 m	MONTH						MONTH
0	1	0	0	0	0	0	0	0	1			
1	2	0	0	0	1	0	0	1	0		-	
		10 Year				YEAR					-	YEAR
0	0	0	0	0	0	0	0	0	0			
9	9	1	0	0	1	0	1	1	0			

Appendix F

LCD Control Data

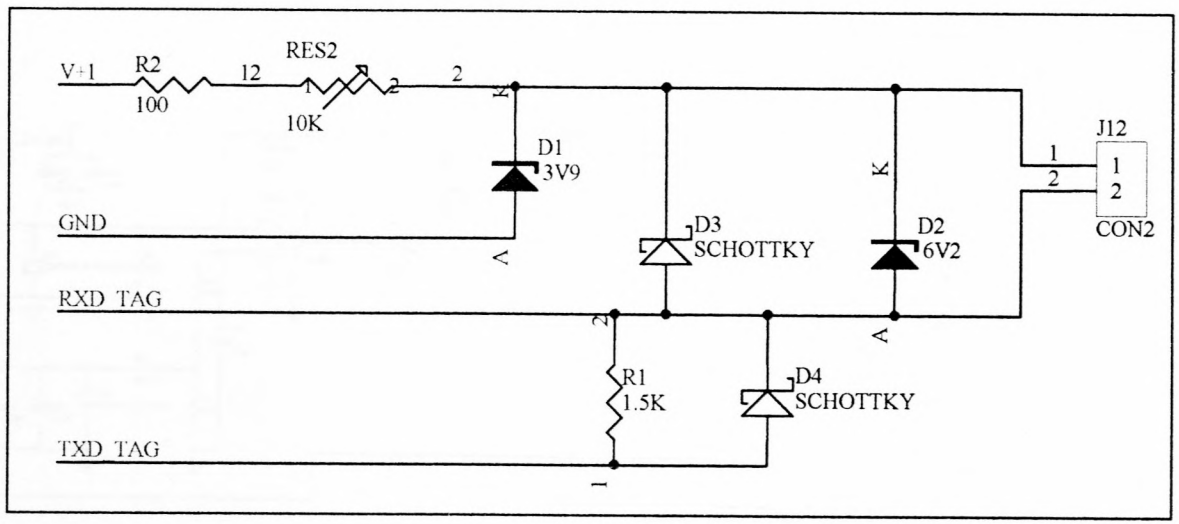


PIN	SIGNAL
1	VSS
2	VDD
3	V0
4	RS
5	R/W
6	E
7	DB0
8	DB1
9	DB2
10	DB3
11	DB4
12	DB5
13	DB6
14	DB7
15	AND
16	CATH

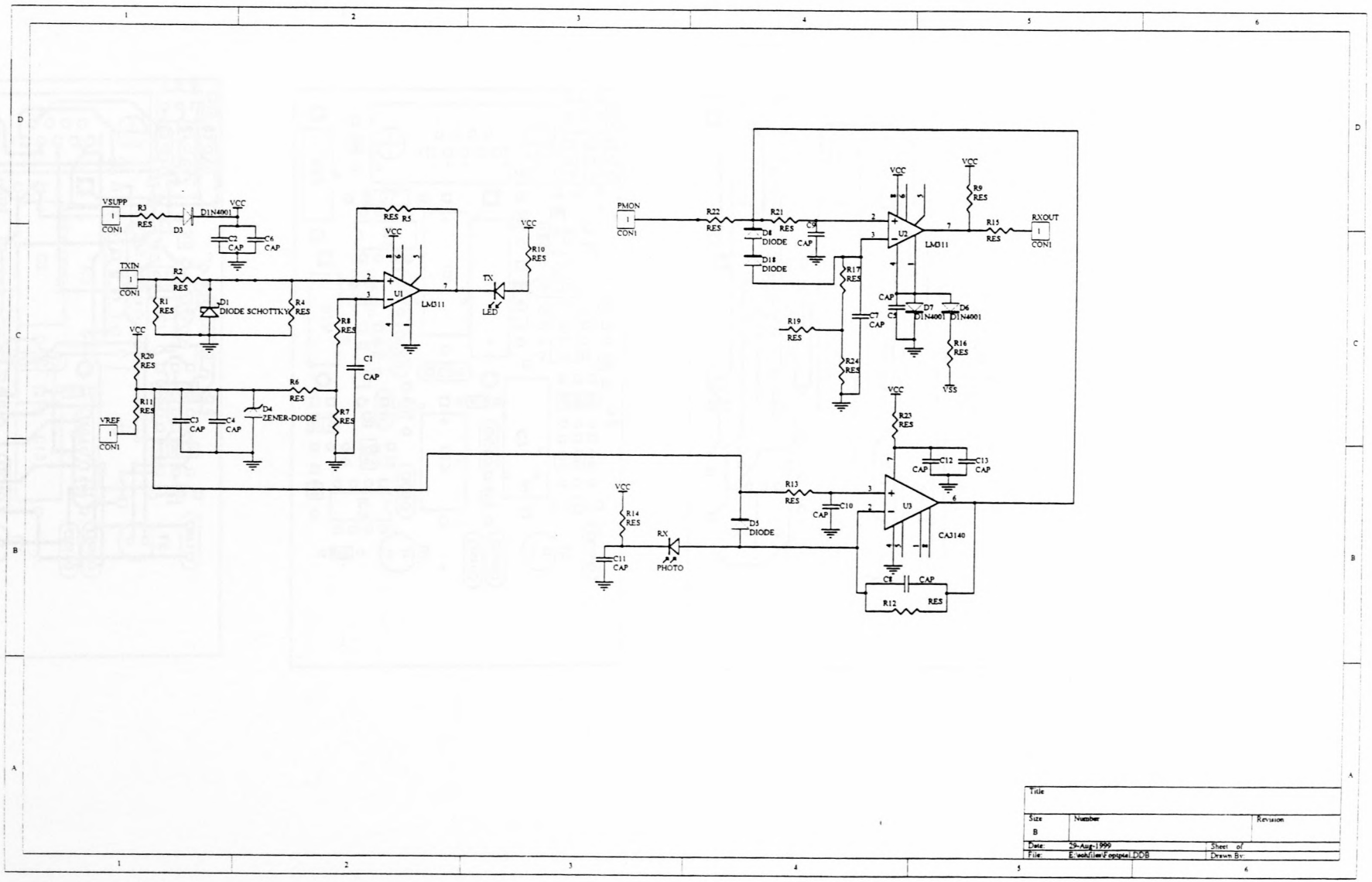
COMMAND	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	EXEC TIME
DISPLAY CLEAR	0	0	0	0	0	0	0	0	0	1	1.64ms
RETURN HOME	0	0	0	0	0	0	0	0	1	X	1.64ms
ENTRY MODE SET	0	0	0	0	0	0	0	1	1-INC 0-DEC	1-CON 0-COFF	40us
DISPLAY ON/OFF	0	0	0	0	0	0	1	1-DON 0-DOFF	1-CON 0-COFF	1-BON 0-BOFF	40us
SHIFT	0	0	0	0	0	1	1-DS 0-CM	1-RSH 0-LSH	X	X	40us
SET FUNCTION	0	0	0	0	1	1-8BIT 0-4BIT	1-2LINE 0-1LINE	1-5X10 0-5X7	X	X	40us
SET CG RAM ADDRESS	0	0	0	1	CG RAM ADDRESS						40us
READ BUSY ADDRESS	0	0	1	DD RAM ADDRESS							40us
READ BUSY FLAG & ADDRESS	0	1	1-BUSY 0-READY	ADDRESS COUNTER FOR BOTH DO & CG RAM ADDRESS							40us
WRITE DATA	1	0	WRITE DATA								40us
READ DATA	1	1	READ DATA								40us

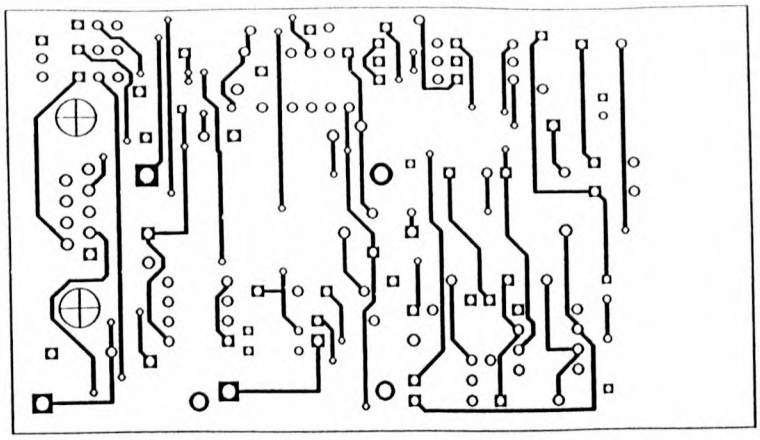
Appendix G

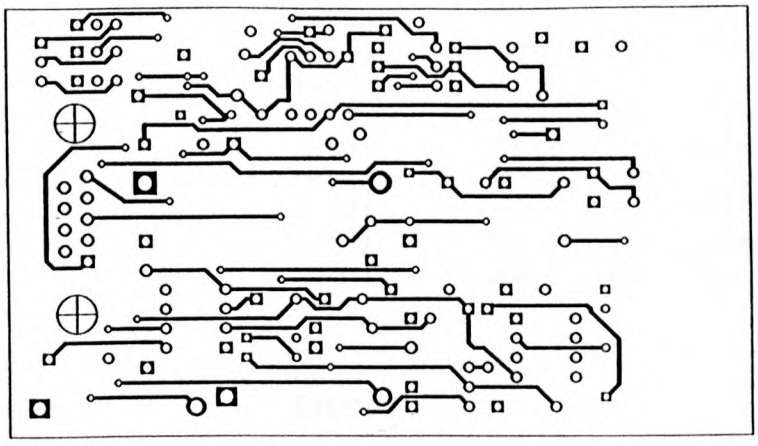
Schematic of Dallas Tag Communication Interface

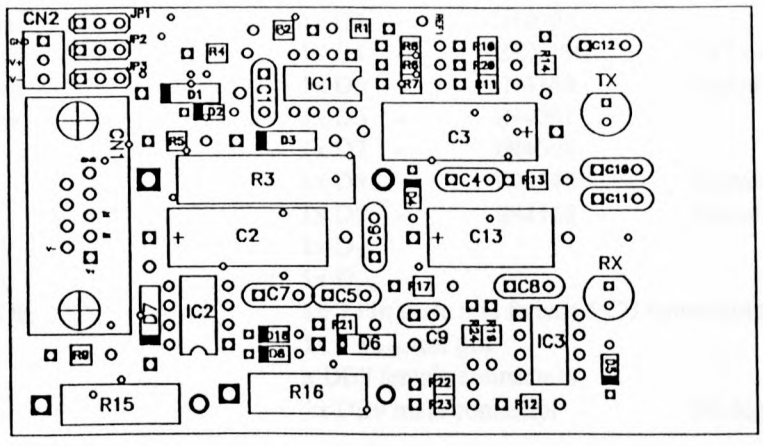


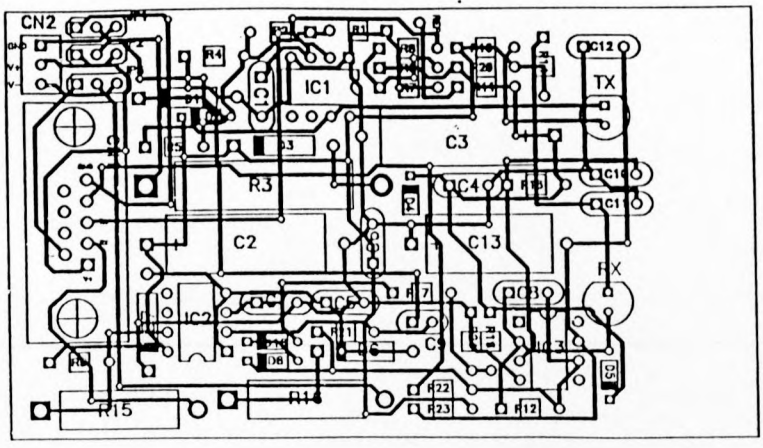
Title: DALLAS TAG COMMUNICATION CIRCUIT			
Size: A4	Number: 1		Revision: 1.2
Date: 12-Dec-1999		Sheet lof 1	
File: D:\temp\daltag.sch		Drawn By: STEVEN SWANEPOEL	


VSUPP
CON1
R3
RES
D3
D1N4001
VCC
C2
CAP
C6
CAP
TXIN
R2
RES
R1
RES
D1
DIODE SCHOTTKY
R4
RES
R5
RES
U1
LM311
TX
LED
R10
RES
R8
RES
C1
CAP
R20
RES
R6
RES
R11
RES
C3
CAP
C4
CAP
D4
ZENER-DIODE
R7
RES
VREF
PMON
R22
RES
R21
RES
C9
CAP
D8
DIODE
D18
DIODE
U2
LM311
R9
RES
R15
RES
RXOUT
R17
RES
R19
RES
C7
CAP
C5
CAP
D7
D1N4001
D6
D1N4001
R16
RES
R24
RES
VSS
R23
RES
C12
CAP
C13
CAP
R13
RES
C10
CAP
U3
CA3140
D5
DIODE
R14
RES
C11
CAP
RX
PHOTO
C8
CAP
R12
RES
Title
Size
B
Number
Revision
Date:
29-Aug-1999
Sheet of
File:
E:\esh\files\Fopspsa1.DDB
Drawn By:






CN2
JP1
IC1
C3
R3
C4
C2
C13
C6
C7
C5
IC2
C9
D6
IC3
R15
R16
C12
TX
C10
C11
RX
C8


CN2
IC1
C3
R3
C4
C2
C13
IC2
R15
C12
TX
RX

Appendix H

Bill of Materials for Optic Fiber Communication Transceiver

RESISTORS

1x R1	-	20K (¼w)
1x R2	-	4K7 (¼w)
1x R3	-	15 (4w)
1x R4	-	20K (¼w)
1x R5	-	120K (¼w)
1x R6	-	6K2 (¼w) 1%
1x R7	-	10K (¼w)
1x R8	-	47K (¼w)
1x R9	-	1K (¼w)
1x R10	-	150 (1w)
1x R11	-	5K1 (¼w)
1x R12	-	1M (¼w)
1x R13	-	100K (¼w)
1x R14	-	27K (¼w)
1x R15	-	47 (2w)
1x R16	-	47 (2w)
1x R17	-	22K (¼w) 1%
1x R18 x		
1x R19	-	150 (¼w)
1x R20	-	3K (¼w)
1x R21	-	3K (¼w)
1x R22	-	10K (¼w)
1x R23	-	340 (¼w)
1x R24	-	4K7 (¼w)

ICs

1x IC1	-	LM311P
1x IC2	-	LM311P
1x IC3	-	CA314OE

1x BCBoard

1x 1 Plastic Box JCD440
120x65x40

CAPACITORS

1x C1	-	100 nF		
1x C2	-	470 μF	(16V)	electrolytic axial
1x C3	-	470 μF	(10V)	electrolytic axial
1x C4	-	100 nF		ceramic
1x C5	-	100 nF		ceramic
1x C6	-	100 nF		ceramic
1x C7	-	100 nF		ceramic
1x C8	-	2,2 pF	(63V)	ceramic
1x C9	-	470 pF	(63V)	ceramic
1x C10	-	100 nF		ceramic
1x C11	-	100 nF		ceramic
1x C12	-	100 nF		ceramic
1x C13	-	200 μF	(16V)	electrolytic axial

DIODES

1x D1	-	BAT49	
1x D2	-	IN4148	Signal diode
1x D3	-	IN4001	
1x D4	-	BZX79C	3V3 or 3,3 Volt
1x D5	-	IN4148	Signal diode
1x D6	-	IN4001	
1x D7	-	IN4001	
1x D8	-	IN4148	Signal diode
1x D9	-	IN4148	Signal diode

1x D_{TX}

1x D_{RX}

1x 3 Pin male and female PCB connectors

1x 1 V_{REF} test pin

x DB9 female connectors

1x DB9 male connector 90 degrees

Appendix I

80C196KC Microcontroller Source Code

All of the embedded source code can be found on the CD appended to the rear cover of volume 2 of this thesis.

Appendix J-1

VHDL Code for PHASE 1 GAL 1 Design

```
-- Binding architecture
--library work;
--use work.cypress.all;
--use work.rtlpkg.all;
library ieee;
use ieee.std_logic_1164.all;

package petgal_xpkg is
        component petgal
        port (clockout, stale, holda, a8, a9, a10, a11, a12, a13, a14, a15, reset : in bit;
                        cs510, ce2, buswidth, waite, ce0, ce1, out0, out1, out2 : out bit;
                        mapp : inout x01z);
        end component;
end  petgal_xpkg;

entity petgal is
        port (clockout, stale, holda, a8, a9, a10, a11, a12, a13, a14, a15, reset : in bit;
                        cs510, ce2, buswidth, waite, ce0, ce1, out0, out1, out2 : out bit;
                        mapp : inout x01z);
end  petgal;

ARCHITECTURE appnote Of c22v10 is

BEGIN
        U1 : petgal PORT MAP (
        clockout => pin1,
        stale => pin2,
        holda => pin3,
        a8 => pin4,
        a9 => pin5,
        a10 => pin6,
        a11 => pin7,
        a12 => pin8,
        a13 => pin9,
        a14 => pin10,
        a15 => pin11,
        reset => pin13,
        fbx(cs510) => pin14,
        fbx(ce2) => pin15,
        fbx(buswidth) => pin16,
        fbx(out0) => pin17,
        fbx(out1) => pin18,
        fbx(waite) => pin19,
        fbx(out2) => pin20,
        fbx(ce0) => pin21,
        fbx(ce1) => pin22,
        mapp => pin23,
        inst => pin24
        );
end appnote;

architecture fsm of petgal is
        signal s0, s1, s2, s3, s4, s5, s6, s7, s8, s9, s10, s11, s12, s13, s14, s16, s17, s18, s19, s20, s21, s22, s23, s24: bit;
        signal nwait_1, nwait_2, nwait_3, nwait_4, nwait_5, nwait_6, nwait_7, ck, mappd1, msig, msig1, waitt : bit;
        signal state:bit_vector (2 downto 0);
        constant async_start: bit_vector (2 downto 0) := "000";
        constant hold_2: bit_vector (2 downto 0) := "001";
        constant hold_3: bit_vector (2 downto 0) := "011";
        constant hold_4: bit_vector (2 downto 0) := "111";
        constant hold_5: bit_vector (2 downto 0) := "110";
        constant hold_6: bit_vector (2 downto 0) := "100";
        constant hold_7: bit_vector (2 downto 0) := "101";
```

```
        constant remove_hold: bit_vector (2 downto 0) := "010";
        constant remove_hold1: bit_vector (2 downto 0) := "000";

begin
        s0      <= (NOT A15) and (NOT A14) and (NOT A13) and (NOT A12);  -- 0000H - 0FFFH
        s1      <= (NOT A15) and (NOT A14) and A13 and (NOT A12);  -- 2000H
        s2      <= (NOT A15) and (NOT A14) and A13 and A12;       -- 3000H
        s3      <= (NOT A15) and A14 and (NOT A13) and (NOT A12);  -- 4000H
        s4      <= (NOT A15) and A14 and (NOT A13) and A12;       -- 5000H
        s5      <= (NOT A15) and A14 and A13 and (NOT A12);       -- 6000H
        s6      <= (NOT A15) and A14 and A13 and A12;            -- 7000H
        s7      <= A15 and (NOT A14) and (NOT A13) and (NOT A12);  -- 8000H
        s8      <= A15 and (NOT A14) and (NOT A13) and A12;       -- 9000H
        s9      <= A15 and (NOT A14) and A13 and (NOT A12);       -- A000H
        s10     <= A15 and (NOT A14) and A13 and A12;          -- B000H
        s11     <= A15 and A14 and (NOT A13) and (NOT A12);     -- C000H
        s12     <= A15 and A14 and (NOT A13) and A12;          -- D000H
        s13     <= A15 and A14 and A13 and (NOT A12);          -- E000H
        s14      <= (A15 and A14 and A13 and A12);            -- F000H
        s15     <= (s0 or s21);
        s16     <= (s1 or s2 or s3 or s4 or s5 or s6 or s7 or s8 or s9 or s10);  -- 2000H - BFFFH 16 bit eprom sim
        s17     <= (NOT A15) and (NOT A14) and (NOT A13) and A12 and A11 and A10 and A9 and (NOT A8); --1E00H -1EFFH UART
        s18     <= (s19 or s17);  -- eprom sim or UART (buswidth)
        s19     <= (s11 or s12 or s13 or s14);  -- C000H - FFFFH 8 bit real time clock nvram
        s20     <= (NOT A15) and (NOT A14) and (NOT A13) and A12 and A11 and A10 and (NOT A9) and (NOT A8); --1C00H -
                1DFFH PERIPH
        s21     <= (NOT A15) and (NOT A14) and (NOT A13) and A12 and (NOT A11) and (NOT A10) and (NOT A9); --1000H -11FFH
                16 - bit RAM
       nwait_1  <= stale and  holda and (nwait_2 or s16); -- 8 bit eprom sim
       nwait_2  <= stale and  holda and (nwait_3 or s17 or s19 or s20);          --NOT A8 SHOULD COME OUT
       nwait_3          <= '0';
        ce2     <= NOT s19;  -- 8 bit real time clock nvram
        ce1     <= NOT s15;  -- 16 bit sram slots for stack
        cs510   <= NOT s17;  -- UART
        buswidth        <= NOT s18;
        ce0     <= NOT s16;  -- 16 bit sim
        mapp    <= '1';
        inst    <= '1';
        waite   <= not waitt;



        s14     <= (A15 and A14 and A13 and A12);          -- F000H
--      s15     <= (s10);
        s16     <= (s1 or s2 or s3 or s4 or s5 or s6 or s7 or s8 or s9);  -- 2000H - AFFFH 16 bit eprom sim
        s17     <= (NOT A15) and (NOT A14) and (NOT A13) and A12 and A11 and A10 and A9 and (NOT A8); --1E00H -1EFFH UART
        s18     <= (s19 or s17);  -- eprom sim or UART (buswidth)
        s19     <= (s10 or s11 or s12 or s13 or s14);  -- B000H - FFFFH 8 bit real time clock nvram
        s20     <= (NOT A15) and (NOT A14) and (NOT A13) and A12 and A11 and A10 and (NOT A9) and (NOT A8); --1C00H -
                1DFFH PERIPH




process(reset, clockout)
begin
                if (reset = '0') then
                        state <= async_start;
                elsif (clockout'event and clockout = '1') then
                        case state is
                                when async_start =>
                                        if (nwait_1 = '1') then
                                                waitt <= '1';
                                        end if;
                                        if (nwait_1 = '1' and nwait_2 = '0') then
                                                state <= remove_hold;
                                        elsif (nwait_2 = '1') then
                                                state <= hold_2;
                                        else
                                                state <= async_start;
                                        end if;
                                when hold_2 =>
                                                waitt <= '1';
                                        if (nwait_3 = '1') then
```

```
                                        state <= hold_3;
                                    else
                                        state <= remove_hold;
                                    end if;
                                when hold_3 =>
                                        waitt <= '1';
                                        state <= remove_hold;
                                when remove_hold =>
                                        waitt <= '0';
                                        state <= async_start;
                                when remove_hold1 =>
                                        waitt <= '0';
                                        state <= async_start;
                                when others =>
                                    state <= async_start;
                            end case;
                        end if;
                end process;
  out0 <= state(0);
out1 <= state(1);
out2 <= state(2);
end fsm;
```

Appendix J-2

VHDL Code for PHASE 1 GAL 2 Design

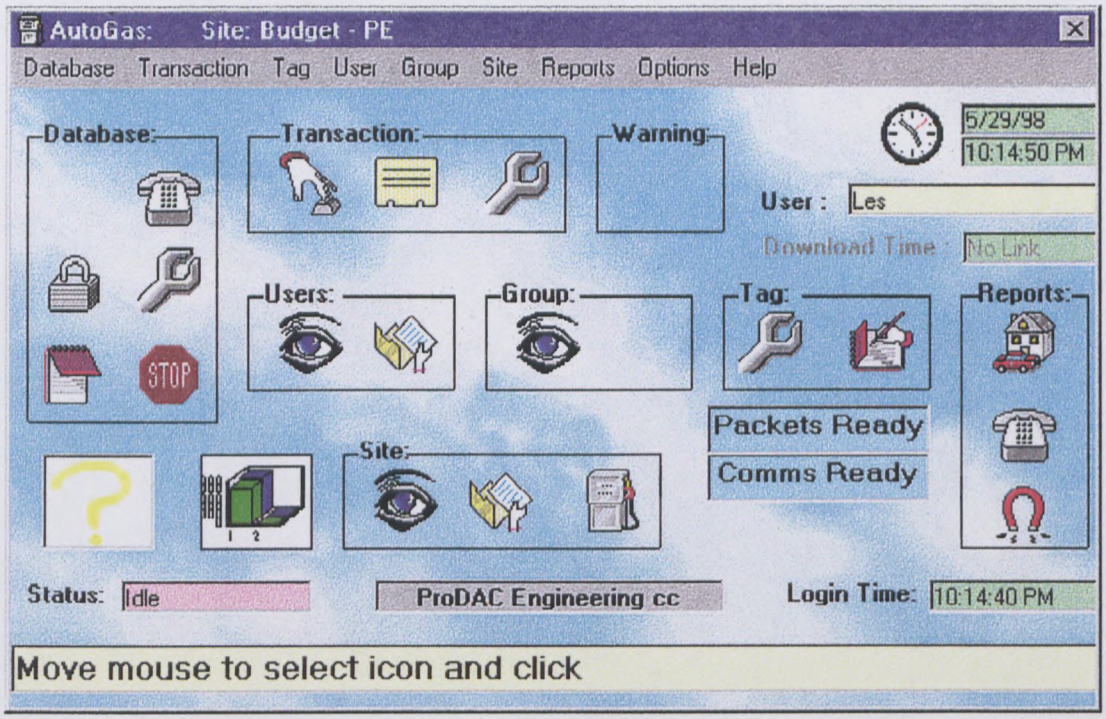
```
-- periph1.vhd

entity PERIPHERAL is
        port(A9      : in bit;
             A10     : in bit;
             WR      : in bit;
             RD      : in bit;
             A15     : in bit;
             A14     : in bit;
             A13     : in bit;
             A12     : in bit;
             A11     : in bit;
             A3      : in bit;
             A2      : in bit;
             A1      : in bit;

             RESA    : out bit;
             RESB    : out bit;
             CELED   : out bit;
             BIT16LED : out bit;
             CE1     : out bit;
             CE2     : out bit;
             CE3     : out bit;
             CE4     : out bit;
             CSRELAY : out bit;
             RES4    : out bit);

attribute pin_numbers of PERIPHERAL:entity is

             "A9:1 " &
             "A10:2  " &
             "A1:3 " &
             "A2:4 " &
             "A3:5 " &
             "WR:6 " &
             "RD:7 " &
             "A11:8 " &
             "A12:9 " &
             "A13:10 " &
             "A14:11 " &
             "A15:13 " &
             "RES4:14 " &
             "CSRELAY:15 " &
             "CE4:16 " &
             "CE3:17 " &
             "CE2:18 " &
             "CE1:19 " &
             "BIT16LED:20 " &
             "CELED:21 " &
             "RESB:22 " &
             "RESA:23";

end PERIPHERAL;

architecture FSM of PERIPHERAL is
signal s1, s2, s3, s4, s5, s6, s7, s8, s9, s10: bit;


begin
        s1      <= ((NOT WR) and ((NOT A15) and (NOT A14) and (NOT A13) and A12 and A11 and A10 and (NOT A9) and A3 and
                (NOT A2) and (A1 or (NOT A1))));
```

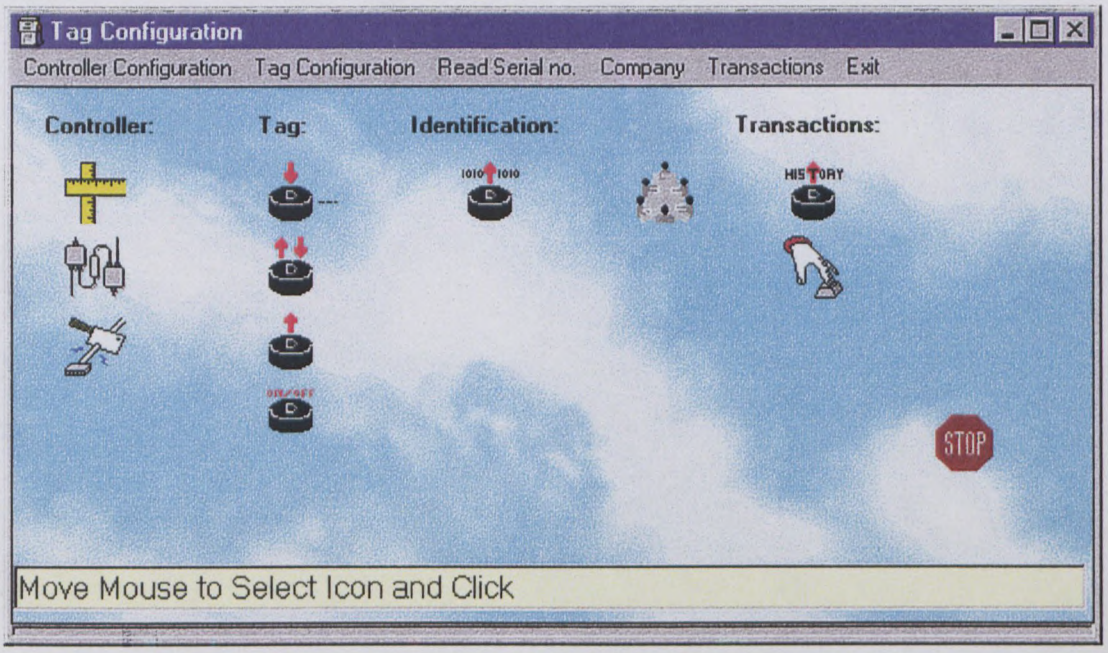
```
        s2          <= ((NOT WR) and ((NOT A15) and (NOT A14) and (NOT A13) and A12 and A11 and A10 and (NOT A9) and A3 and ((A2
                    and (NOT A1))  or ((NOT A2) and A1))));
        RESA        <= (NOT s1);
        RESB        <= (NOT s2);
        CE4         <= (NOT(((NOT A15) and (NOT A14) and (NOT A13) and A12 and A11 and A10 and (NOT A9) and (NOT A3) and  (NOT
                    A2) and (NOT A1)) and (NOT RD)));
        CE3         <= (NOT(((NOT A15) and (NOT A14) and (NOT A13) and A12 and A11 and A10 and (NOT A9) and (NOT A3) and (NOT
                    A2) and A1) and (NOT RD)));
        CE2         <= (NOT(((NOT A15) and (NOT A14) and (NOT A13) and A12 and A11 and A10 and (NOT A9) and (NOT A3) and A2 and
                    (NOT A1)) and (NOT RD)));
        CE1         <= (NOT(((NOT A15) and (NOT A14) and (NOT A13) and A12 and A11 and A10 and (NOT A9) and (NOT A3) and A2 and
                    A1) and (NOT RD)));
        RES4        <= ((NOT WR) and (((NOT A15) and (NOT A14) and (NOT A13) and A12 and A11 and A10 and (NOT A9) and (NOT A3)
                    and (NOT A2) and A1)));
      BIT16LED <= ((NOT WR) and (((NOT A15) and (NOT A14) and (NOT A13) and A12 and A11 and A10 and (NOT A9) and (NOT A3)
                    and A2 and (NOT A1))));
        CELED   <= ((NOT WR) and (((NOT A15) and (NOT A14) and (NOT A13) and A12 and A11 and A10 and (NOT A9) and (NOT A3)
                    and A2 and A1)));
        CSRELAY <= ((NOT WR) and (((NOT A15) and (NOT A14) and (NOT A13) and A12 and A11 and A10 and (NOT A9) and (NOT A3)
                    and (NOT A2) and (NOT A1))));

end FSM;
```

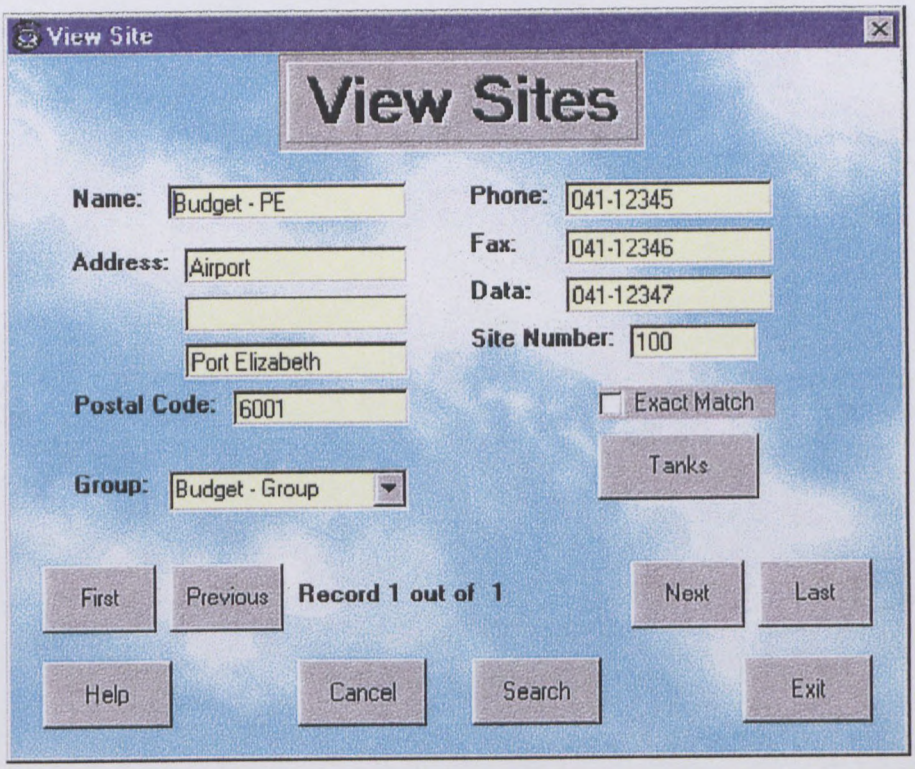
Appendix K

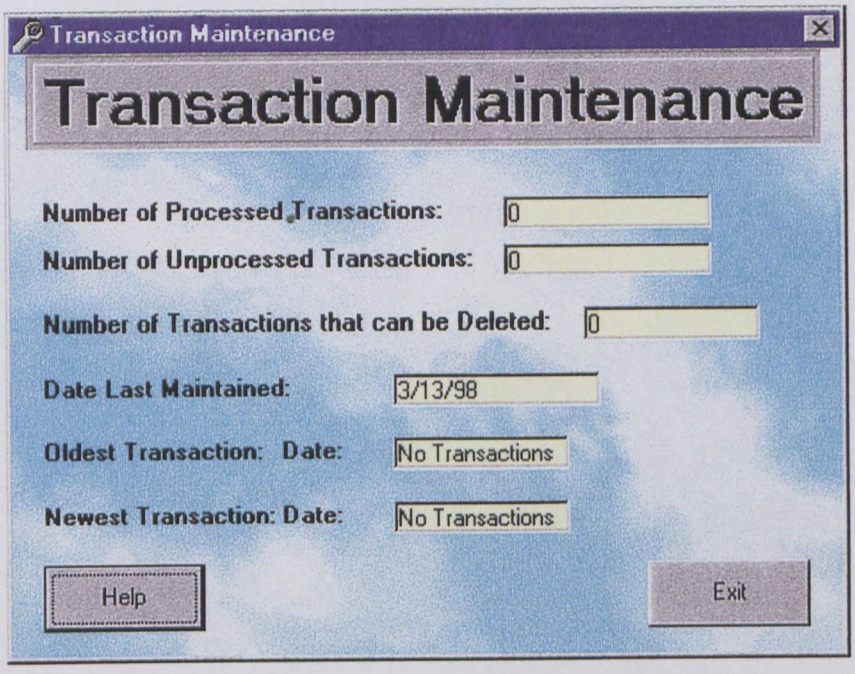
Host Machine Communication Software

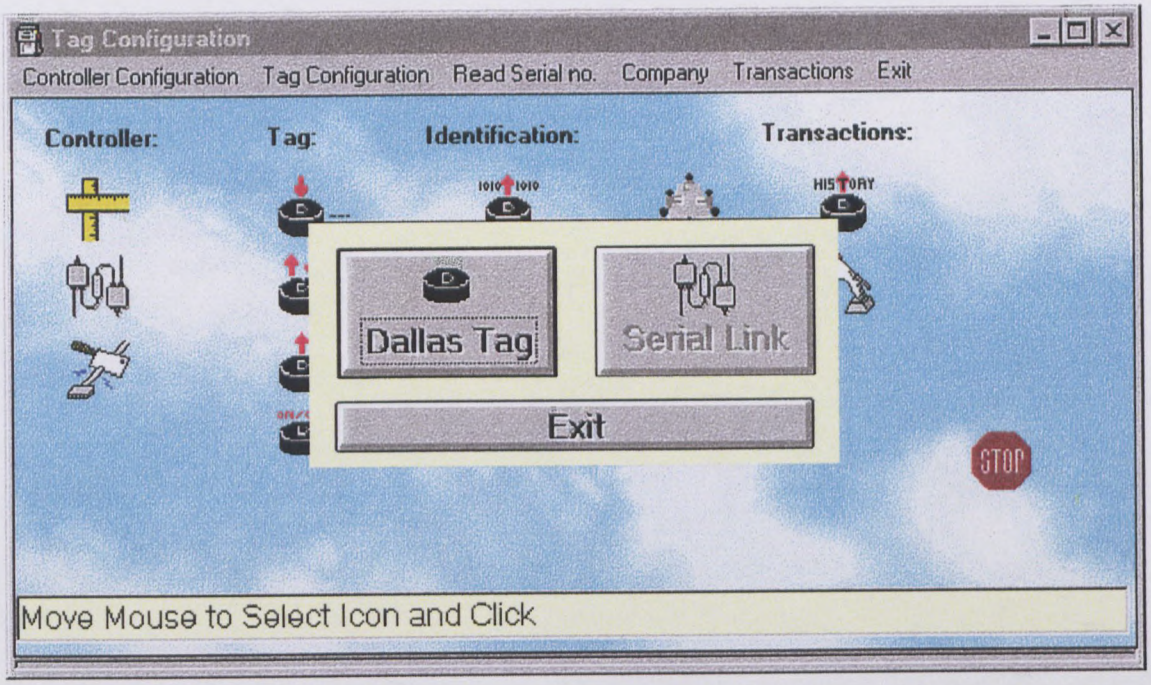
All of the source code for the forms presented in this appendix, can be found on the CD appended to the rear cover of volume 2 of this thesis.

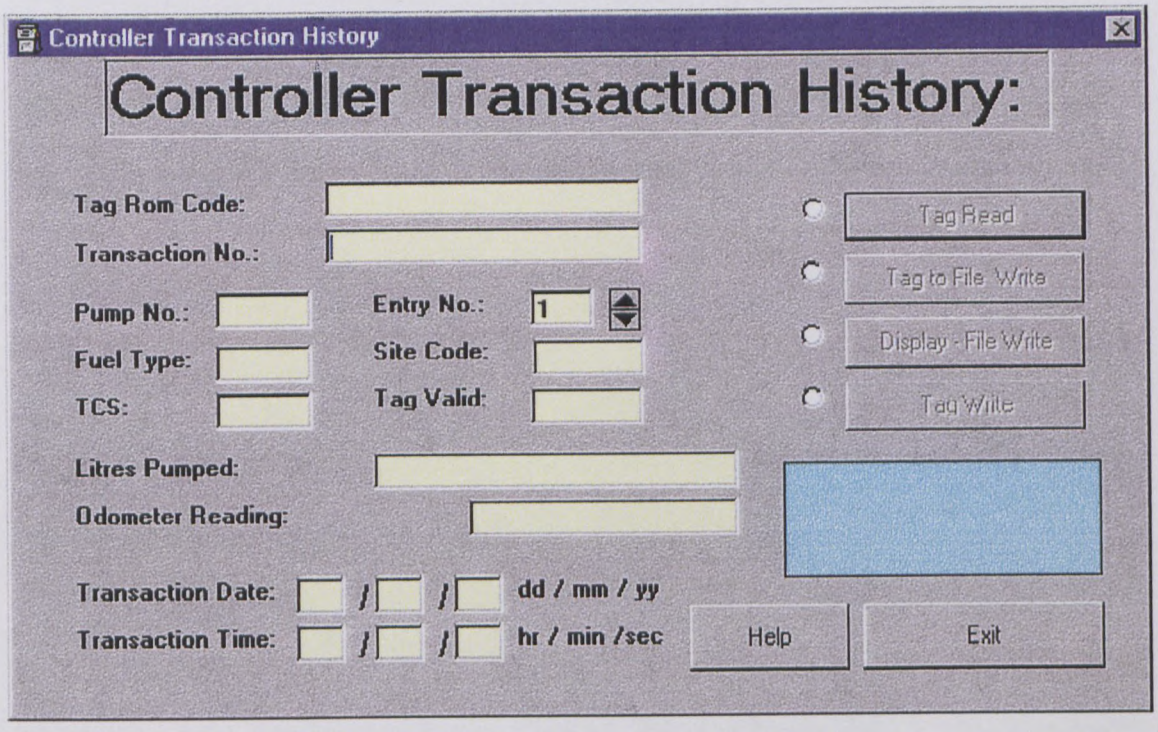

AutoGas: Site: Budget - PE
Database Transaction Tag User Group Site Reports Options Help
Database:
Transaction:
Warning
5/29/98
10:14:50 PM
User : Les
Download Time : No Link
Users:
Group:
Tag:
Reports:
STOP
Packets Ready
Comms Ready
Site:
Status: Idle
ProDAC Engineering cc
Login Time: 10:14:40 PM
Move mouse to select icon and click

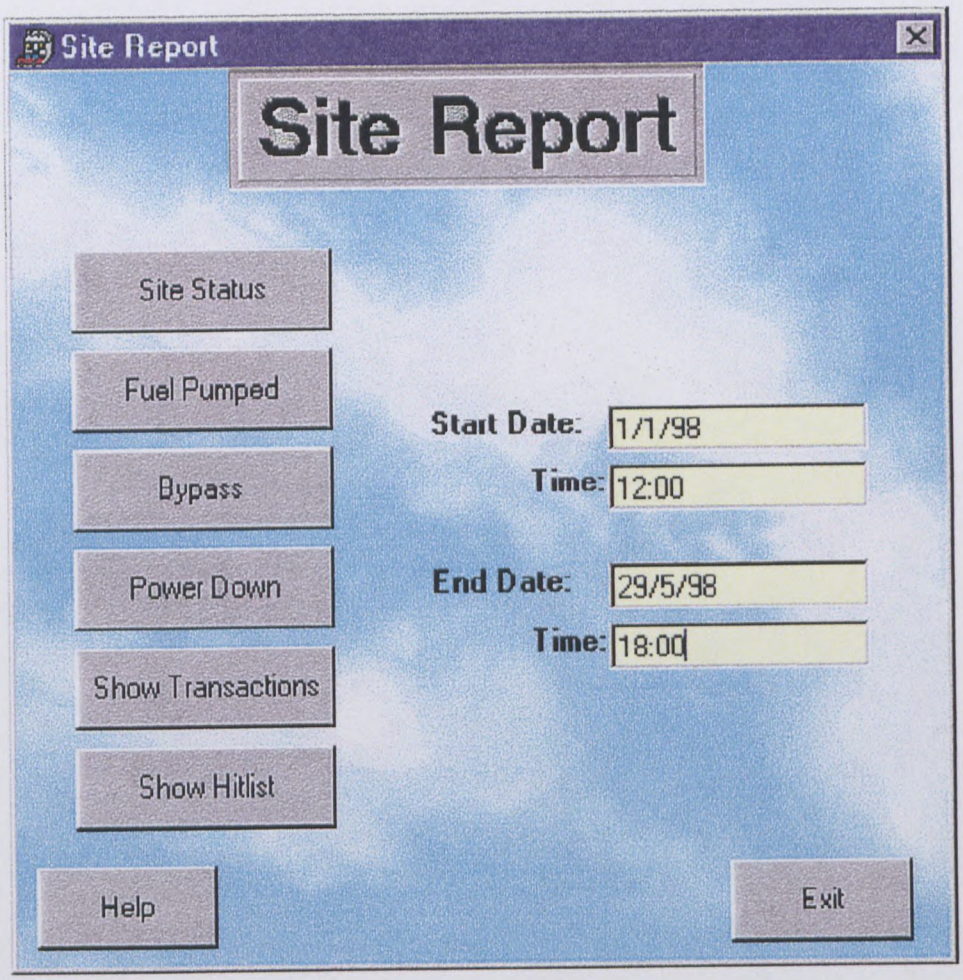

Tag Configuration
Controller Configuration Tag Configuration Read Serial no. Company Transactions Exit
Controller:
Tag:
Identification:
Transactions:
HISTORY
STOP
Move Mouse to Select Icon and Click

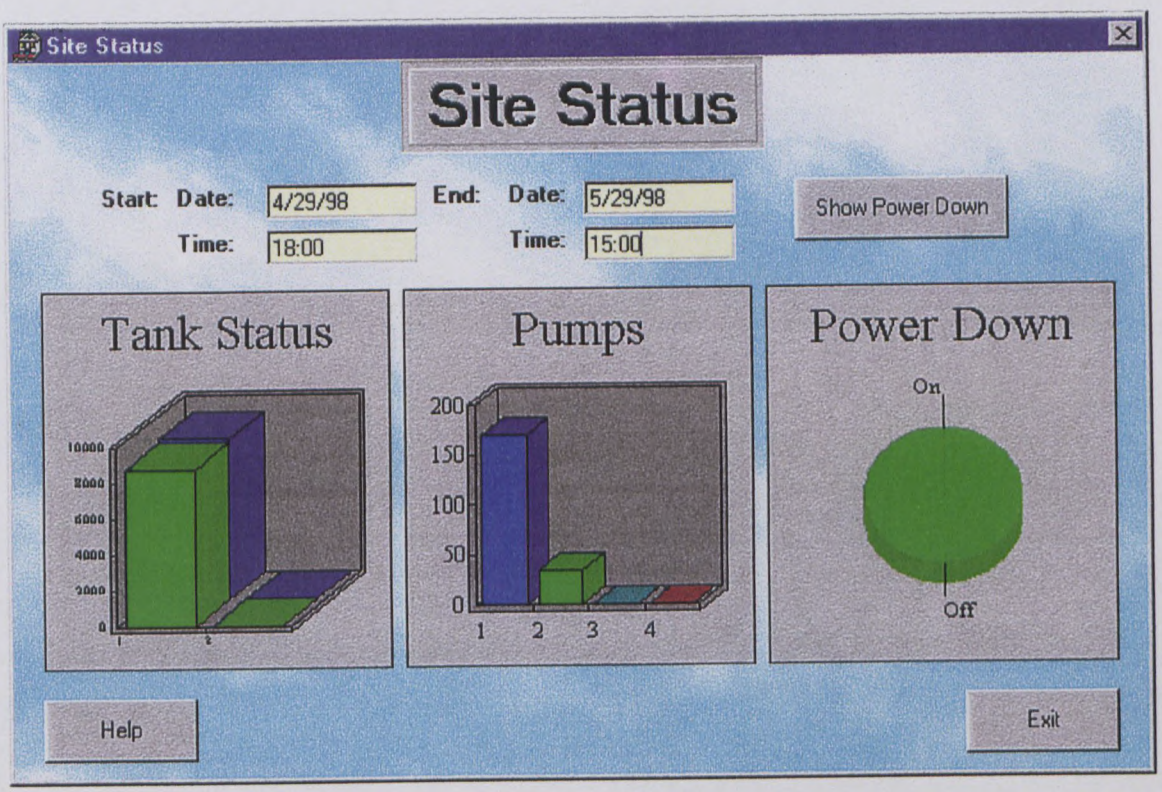

View Site
View Sites
Name:
Budget - PE
Address:
Airport
Port Elizabeth
Postal Code:
6001
Group:
Budget - Group
Phone:
041-12345
Fax:
041-12346
Data:
041-12347
Site Number:
100
Exact Match
Tanks
First
Previous
Record 1 out of 1
Next
Last
Help
Cancel
Search
Exit

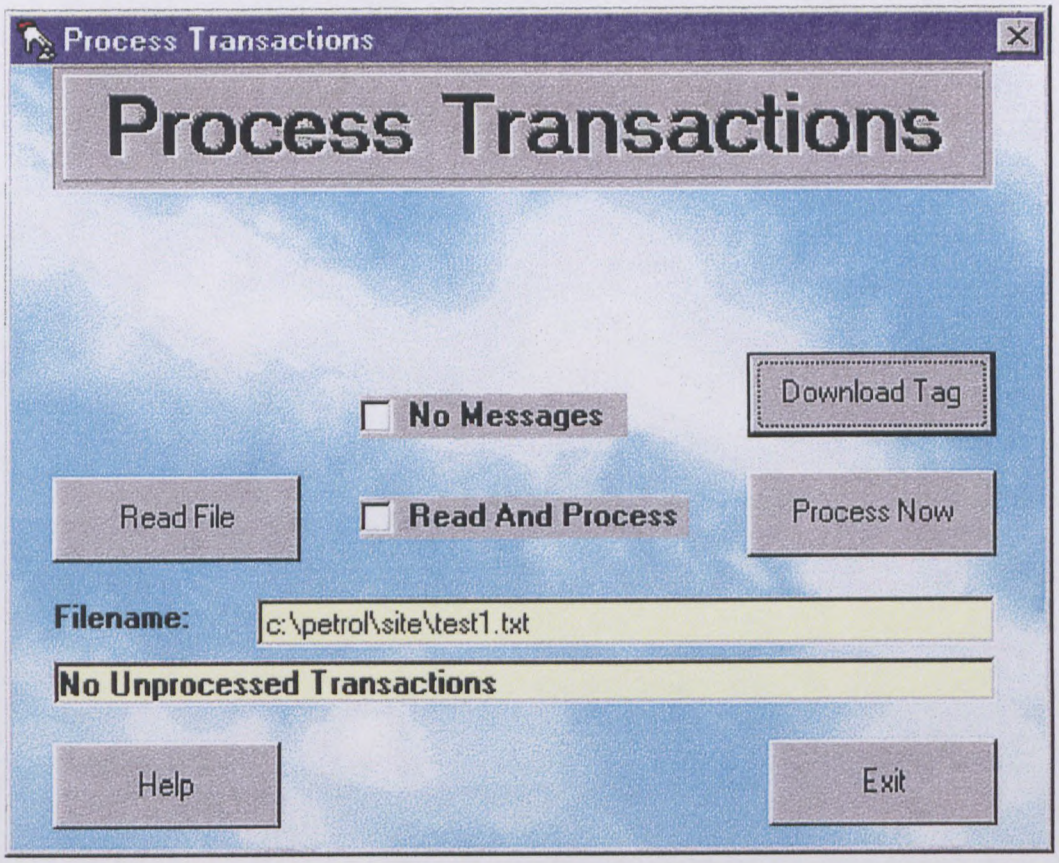

Transaction Maintenance
Transaction Maintenance
Number of Processed Transactions:
0
Number of Unprocessed Transactions:
0
Number of Transactions that can be Deleted:
0
Date Last Maintained:
3/13/98
Oldest Transaction: Date:
No Transactions
Newest Transaction: Date:
No Transactions
Help
Exit

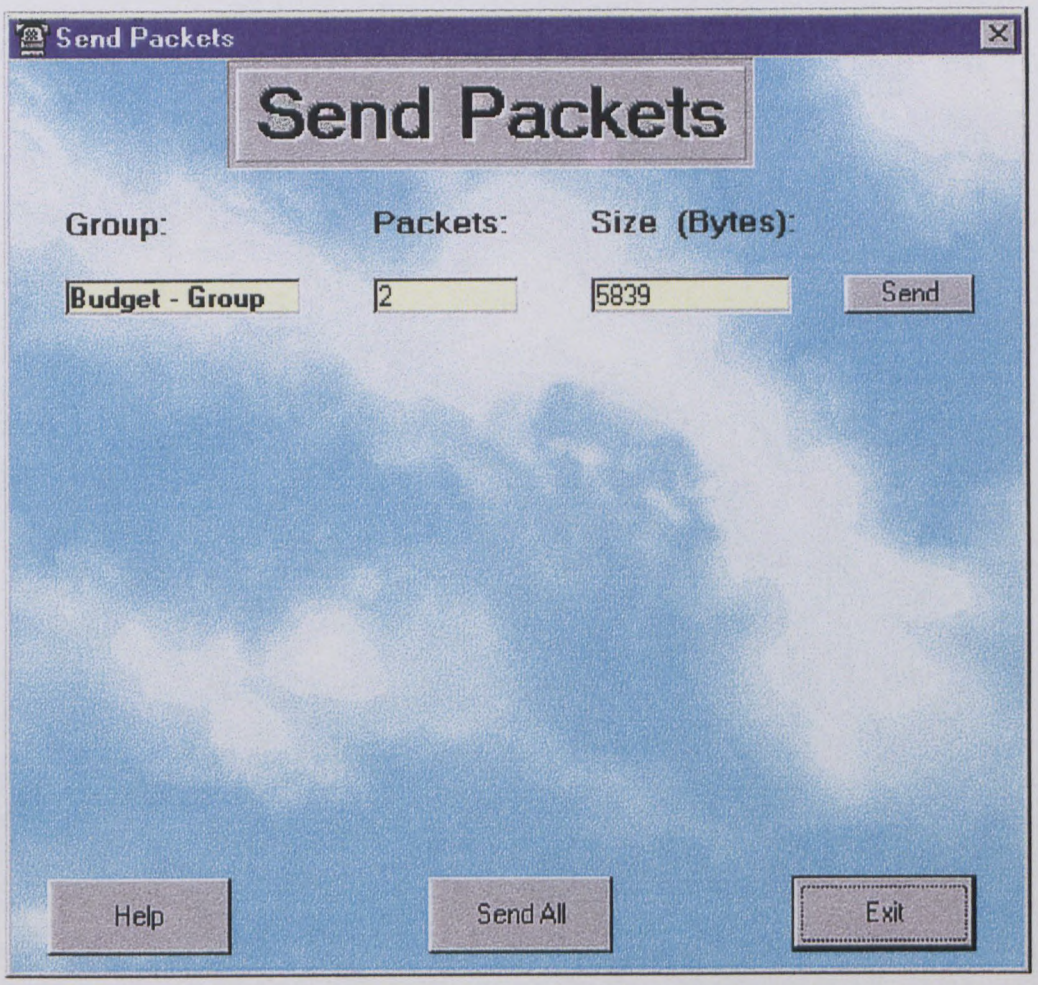

Tag Configuration
Controller Configuration
Tag Configuration
Read Serial no.
Company
Transactions
Exit
Controller:
Tag:
Identification:
Transactions:
Dallas Tag
Serial Link
Exit
STOP
Move Mouse to Select Icon and Click

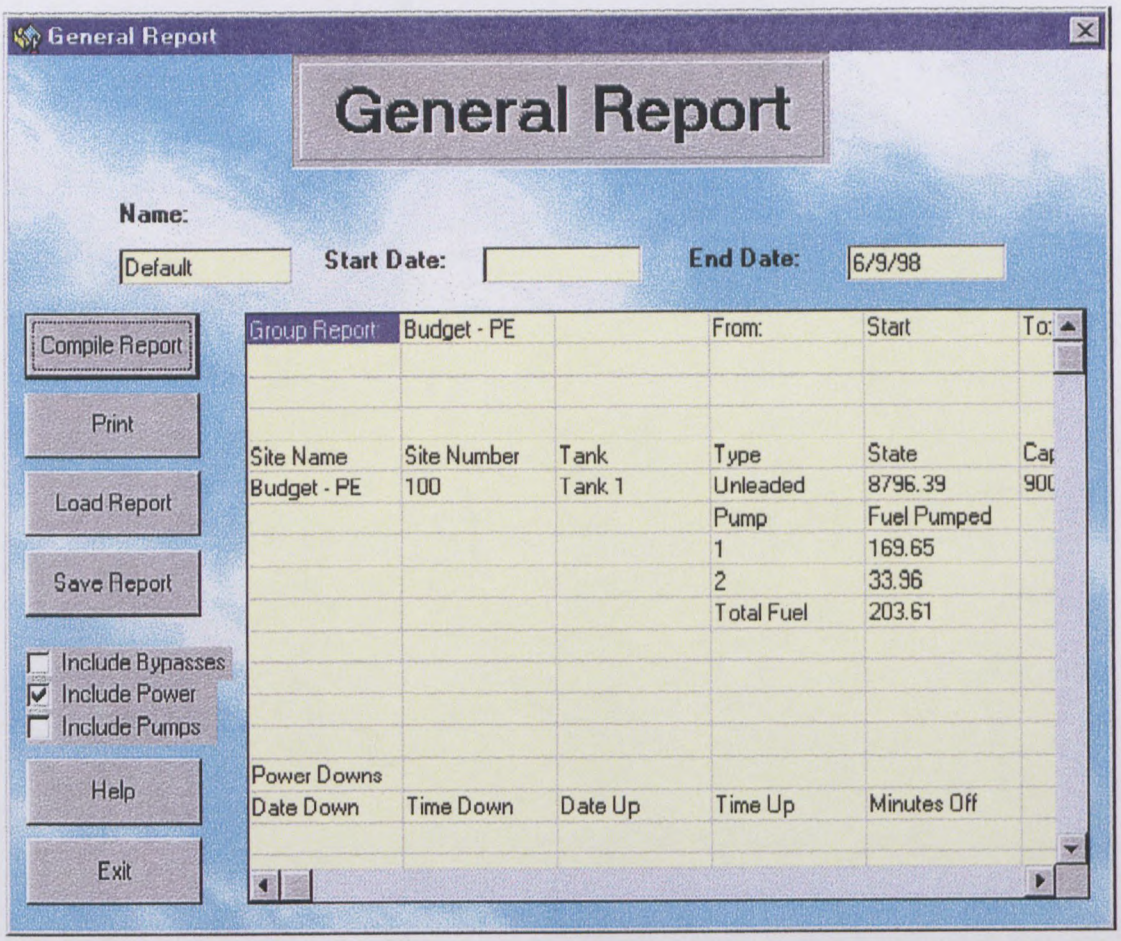

Controller Transaction History
Controller Transaction History:
Tag Rom Code:
Transaction No.:
Pump No.:
Entry No.:
1
Fuel Type:
Site Code:
TCS:
Tag Valid:
Litres Pumped:
Odometer Reading:
Transaction Date:
dd / mm / yy
Transaction Time:
hr / min /sec
Tag Read
Tag to File Write
Display - File Write
Tag Write
Help
Exit

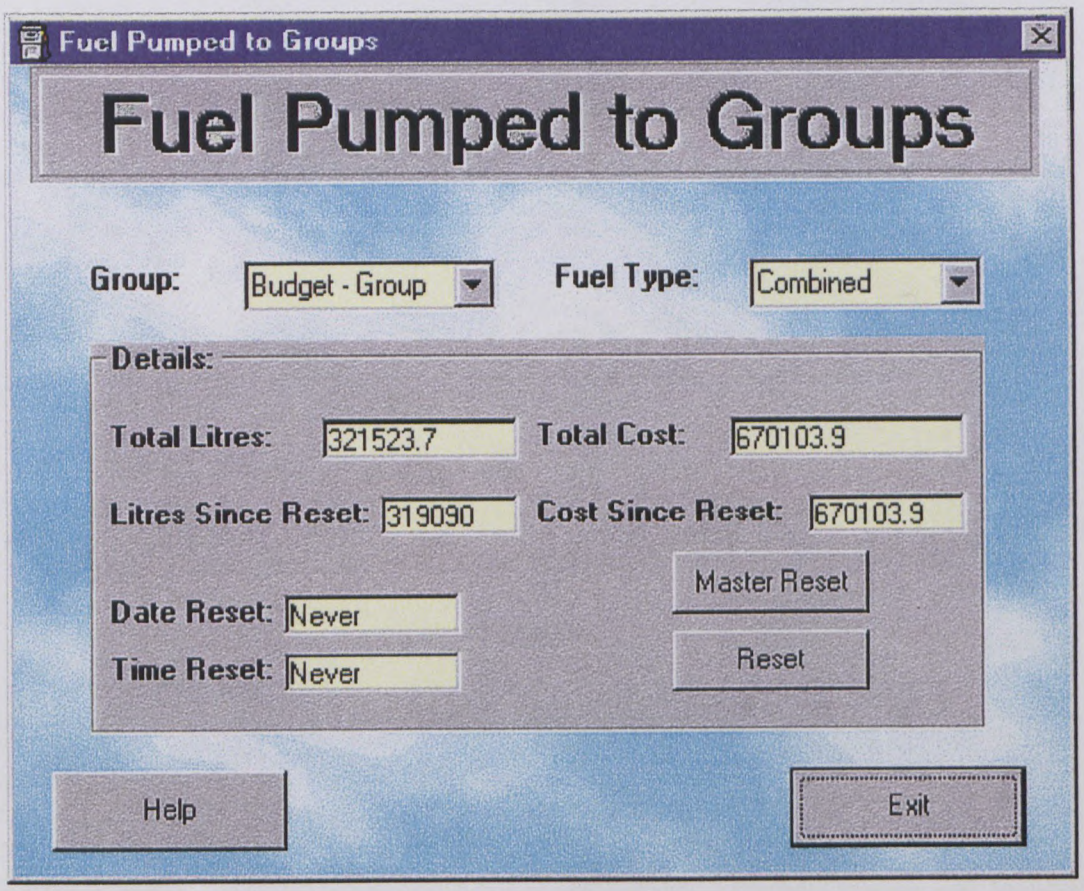

Site Report
Site Report
Site Status
Fuel Pumped
Bypass
Power Down
Show Transactions
Show Hitlist
Start Date: 1/1/98
Time: 12:00
End Date: 29/5/98
Time: 18:00
Help
Exit

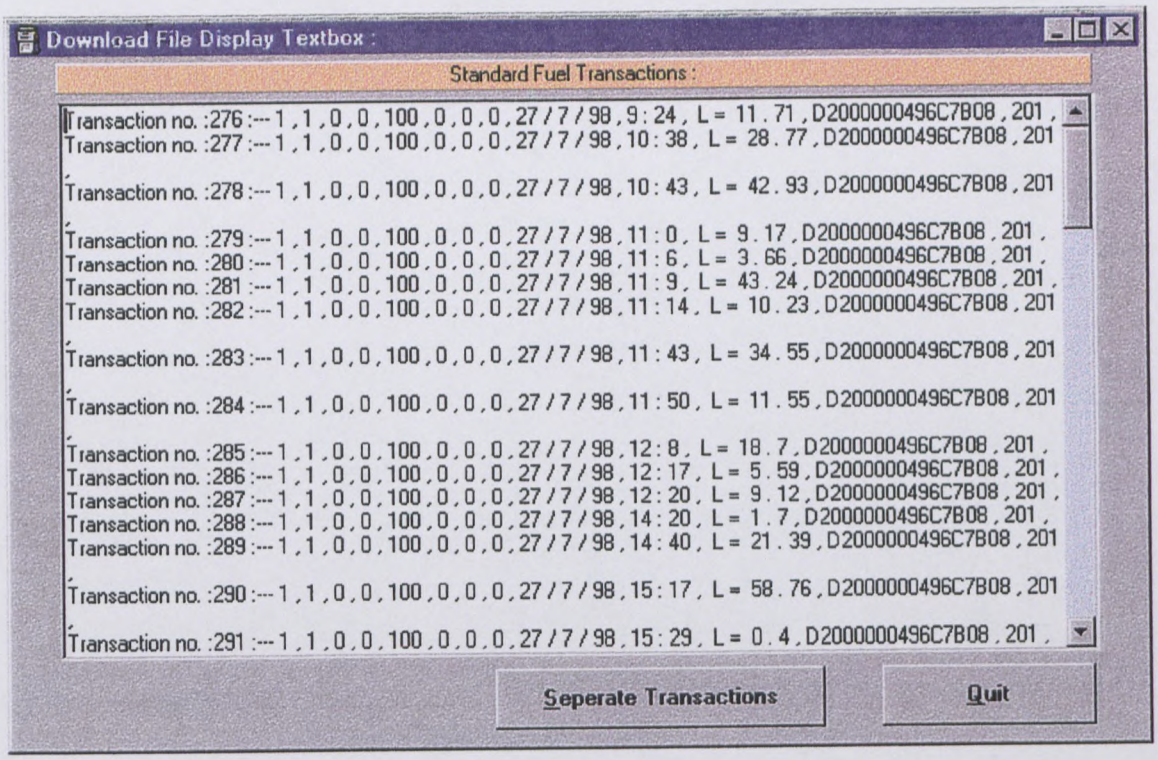

Site Status
Site Status
Start: Date: 4/29/98
Time: 18:00
End: Date: 5/29/98
Time: 15:00
Show Power Down
Tank Status
Pumps
200
150
100
50
0
1 2 3 4
Power Down
On
Off
Help
Exit

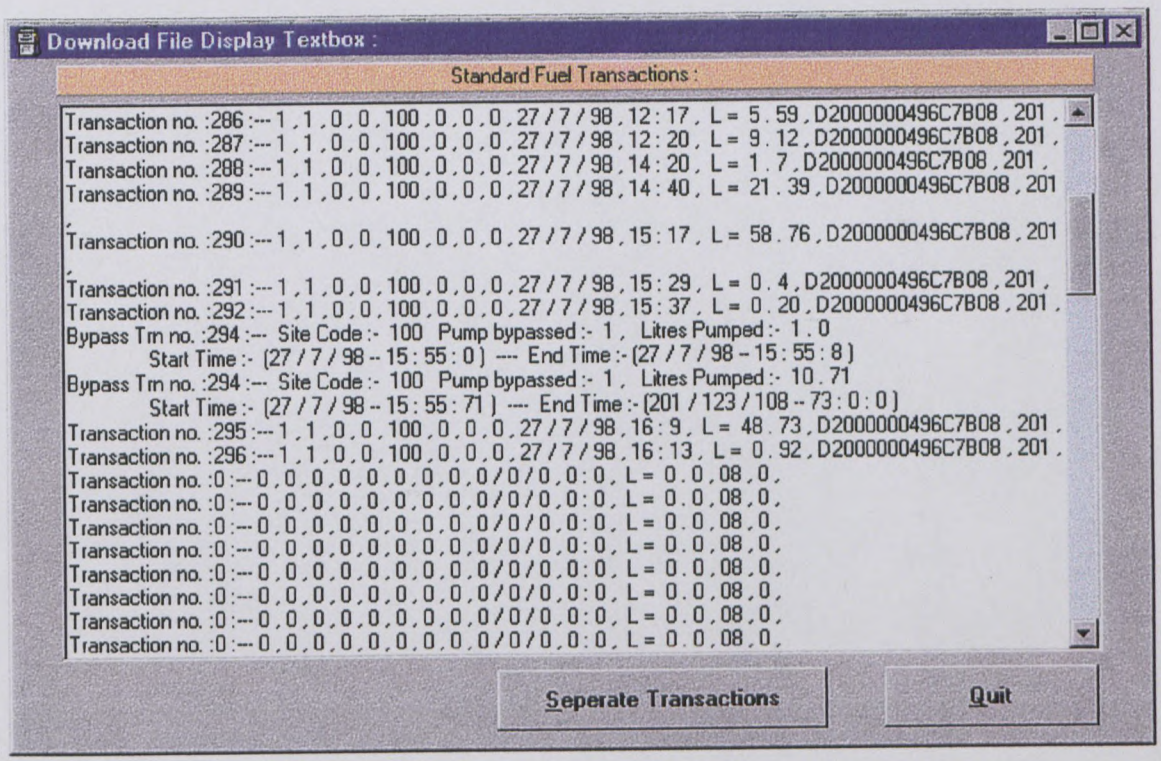

Process Transactions
Process Transactions
No Messages
Download Tag
Read File
Read And Process
Process Now
Filename:
c:\petrol\site\test1.txt
No Unprocessed Transactions
Help
Exit

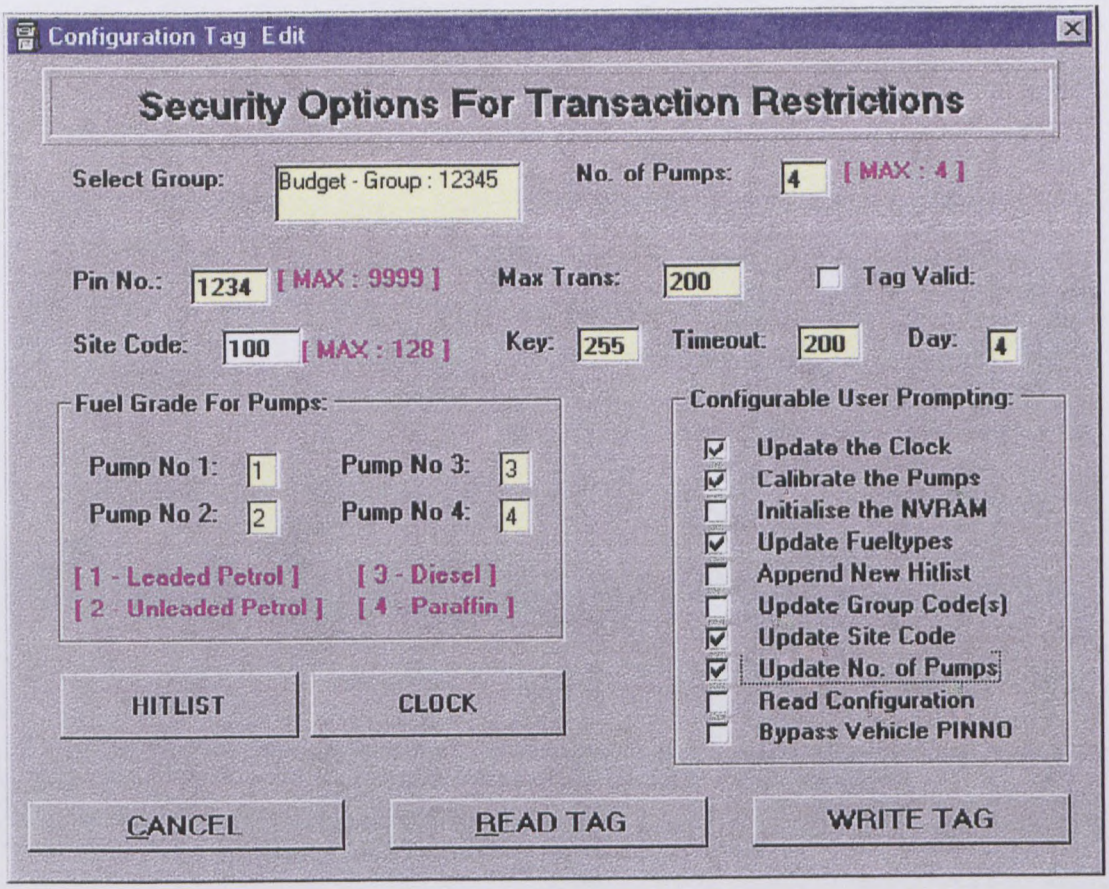

Send Packets
Send Packets
Group:
Packets:
Size (Bytes):
Budget - Group
2
5839
Send
Help
Send All
Exit

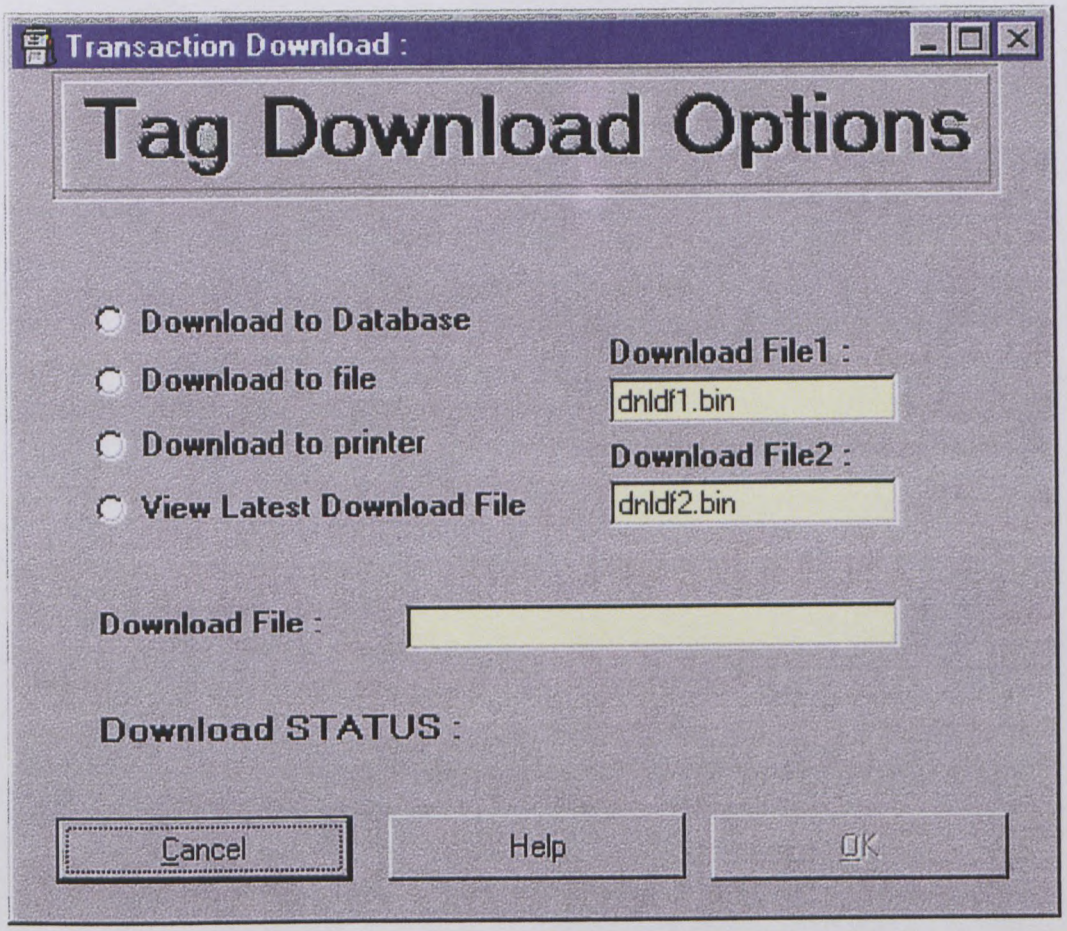

General Report
General Report
Name:
Default
Start Date:
End Date:
6/9/98
Compile Report
Print
Load Report
Save Report
Include Bypasses
Include Power
Include Pumps
Help
Exit
Group Report
Budget - PE
From:
Start
To:
Site Name
Site Number
Tank
Type
State
Budget - PE
100
Tank 1
Unleaded
8796.39
Pump
Fuel Pumped
1
169.65
2
33.96
Total Fuel
203.61
Power Downs
Date Down
Time Down
Date Up
Time Up
Minutes Off


Fuel Pumped to Groups
Fuel Pumped to Groups
Group:
Budget - Group
Fuel Type:
Combined
Details:
Total Litres:
321523.7
Total Cost:
670103.9
Litres Since Reset:
319090
Cost Since Reset:
670103.9
Master Reset
Date Reset:
Never
Time Reset:
Never
Reset
Help
Exit


Download File Display Textbox :
Standard Fuel Transactions :
Transaction no. :276 :--- 1 , 1 , 0 , 0 , 100 , 0 , 0 , 0 , 27 / 7 / 98 , 9 : 24 , L = 11 . 71 , D2000000496C7B08 , 201 ,
Transaction no. :277 :--- 1 , 1 , 0 , 0 , 100 , 0 , 0 , 0 , 27 / 7 / 98 , 10 : 38 , L = 28 . 77 , D2000000496C7B08 , 201
,
Transaction no. :278 :--- 1 , 1 , 0 , 0 , 100 , 0 , 0 , 0 , 27 / 7 / 98 , 10 : 43 , L = 42 . 93 , D2000000496C7B08 , 201
,
Transaction no. :279 :--- 1 , 1 , 0 , 0 , 100 , 0 , 0 , 0 , 27 / 7 / 98 , 11 : 0 , L = 9 . 17 , D2000000496C7B08 , 201 ,
Transaction no. :280 :--- 1 , 1 , 0 , 0 , 100 , 0 , 0 , 0 , 27 / 7 / 98 , 11 : 6 , L = 3 . 66 , D2000000496C7B08 , 201 ,
Transaction no. :281 :--- 1 , 1 , 0 , 0 , 100 , 0 , 0 , 0 , 27 / 7 / 98 , 11 : 9 , L = 43 . 24 , D2000000496C7B08 , 201 ,
Transaction no. :282 :--- 1 , 1 , 0 , 0 , 100 , 0 , 0 , 0 , 27 / 7 / 98 , 11 : 14 , L = 10 . 23 , D2000000496C7B08 , 201
,
Transaction no. :283 :--- 1 , 1 , 0 , 0 , 100 , 0 , 0 , 0 , 27 / 7 / 98 , 11 : 43 , L = 34 . 55 , D2000000496C7B08 , 201
,
Transaction no. :284 :--- 1 , 1 , 0 , 0 , 100 , 0 , 0 , 0 , 27 / 7 / 98 , 11 : 50 , L = 11 . 55 , D2000000496C7B08 , 201
,
Transaction no. :285 :--- 1 , 1 , 0 , 0 , 100 , 0 , 0 , 0 , 27 / 7 / 98 , 12 : 8 , L = 18 . 7 , D2000000496C7B08 , 201 ,
Transaction no. :286 :--- 1 , 1 , 0 , 0 , 100 , 0 , 0 , 0 , 27 / 7 / 98 , 12 : 17 , L = 5 . 59 , D2000000496C7B08 , 201 ,
Transaction no. :287 :--- 1 , 1 , 0 , 0 , 100 , 0 , 0 , 0 , 27 / 7 / 98 , 12 : 20 , L = 9 . 12 , D2000000496C7B08 , 201 ,
Transaction no. :288 :--- 1 , 1 , 0 , 0 , 100 , 0 , 0 , 0 , 27 / 7 / 98 , 14 : 20 , L = 1 . 7 , D2000000496C7B08 , 201 ,
Transaction no. :289 :--- 1 , 1 , 0 , 0 , 100 , 0 , 0 , 0 , 27 / 7 / 98 , 14 : 40 , L = 21 . 39 , D2000000496C7B08 , 201
,
Transaction no. :290 :--- 1 , 1 , 0 , 0 , 100 , 0 , 0 , 0 , 27 / 7 / 98 , 15 : 17 , L = 58 . 76 , D2000000496C7B08 , 201
,
Transaction no. :291 :--- 1 , 1 , 0 , 0 , 100 , 0 , 0 , 0 , 27 / 7 / 98 , 15 : 29 , L = 0 . 4 , D2000000496C7B08 , 201 ,
Seperate Transactions
Quit


Download File Display Textbox :
Standard Fuel Transactions :
Transaction no. :286 :--- 1 , 1 , 0 , 0 , 100 , 0 , 0 , 0 , 27 / 7 / 98 , 12 : 17 , L = 5 . 59 , D2000000496C7B08 , 201 ,
Transaction no. :287 :--- 1 , 1 , 0 , 0 , 100 , 0 , 0 , 0 , 27 / 7 / 98 , 12 : 20 , L = 9 . 12 , D2000000496C7B08 , 201 ,
Transaction no. :288 :--- 1 , 1 , 0 , 0 , 100 , 0 , 0 , 0 , 27 / 7 / 98 , 14 : 20 , L = 1 . 7 , D2000000496C7B08 , 201 ,
Transaction no. :289 :--- 1 , 1 , 0 , 0 , 100 , 0 , 0 , 0 , 27 / 7 / 98 , 14 : 40 , L = 21 . 39 , D2000000496C7B08 , 201
,
Transaction no. :290 :--- 1 , 1 , 0 , 0 , 100 , 0 , 0 , 0 , 27 / 7 / 98 , 15 : 17 , L = 58 . 76 , D2000000496C7B08 , 201
,
Transaction no. :291 :--- 1 , 1 , 0 , 0 , 100 , 0 , 0 , 0 , 27 / 7 / 98 , 15 : 29 , L = 0 . 4 , D2000000496C7B08 , 201 ,
Transaction no. :292 :--- 1 , 1 , 0 , 0 , 100 , 0 , 0 , 0 , 27 / 7 / 98 , 15 : 37 , L = 0 . 20 , D2000000496C7B08 , 201 ,
Bypass Trn no. :294 :--- Site Code :- 100 Pump bypassed :- 1 , Litres Pumped :- 1 . 0
Start Time :- (27 / 7 / 98 -- 15 : 55 : 0) --- End Time :- (27 / 7 / 98 -- 15 : 55 : 8)
Bypass Trn no. :294 :--- Site Code :- 100 Pump bypassed :- 1 , Litres Pumped :- 10 . 71
Start Time :- (27 / 7 / 98 -- 15 : 55 : 71) --- End Time :- (201 / 123 / 108 -- 73 : 0 : 0)
Transaction no. :295 :--- 1 , 1 , 0 , 0 , 100 , 0 , 0 , 0 , 27 / 7 / 98 , 16 : 9 , L = 48 . 73 , D2000000496C7B08 , 201 ,
Transaction no. :296 :--- 1 , 1 , 0 , 0 , 100 , 0 , 0 , 0 , 27 / 7 / 98 , 16 : 13 , L = 0 . 92 , D2000000496C7B08 , 201 ,
Transaction no. :0 :--- 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 / 0 / 0 , 0 : 0 , L = 0 . 0 , 08 , 0 ,
Transaction no. :0 :--- 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 / 0 / 0 , 0 : 0 , L = 0 . 0 , 08 , 0 ,
Transaction no. :0 :--- 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 / 0 / 0 , 0 : 0 , L = 0 . 0 , 08 , 0 ,
Transaction no. :0 :--- 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 / 0 / 0 , 0 : 0 , L = 0 . 0 , 08 , 0 ,
Transaction no. :0 :--- 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 / 0 / 0 , 0 : 0 , L = 0 . 0 , 08 , 0 ,
Transaction no. :0 :--- 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 / 0 / 0 , 0 : 0 , L = 0 . 0 , 08 , 0 ,
Transaction no. :0 :--- 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 / 0 / 0 , 0 : 0 , L = 0 . 0 , 08 , 0 ,
Seperate Transactions
Quit


Configuration Tag Edit
Security Options For Transaction Restrictions
Select Group:
Budget - Group : 12345
No. of Pumps:
4
[MAX : 4]
Pin No.:
1234
[MAX : 9999]
Max Trans:
200
Tag Valid:
Site Code:
100
[MAX : 128]
Key:
255
Timeout:
200
Day:
4
Fuel Grade For Pumps:
Pump No 1:
1
Pump No 3:
3
Pump No 2:
2
Pump No 4:
4
[1 - Leaded Petrol]
[3 - Diesel]
[2 - Unleaded Petrol]
[4 - Paraffin]
HITLIST
CLOCK
Configurable User Prompting:
Update the Clock
Calibrate the Pumps
Initialise the NVRAM
Update Fueltypes
Append New Hitlist
Update Group Code(s)
Update Site Code
Update No. of Pumps
Read Configuration
Bypass Vehicle PINNO
CANCEL
READ TAG
WRITE TAG

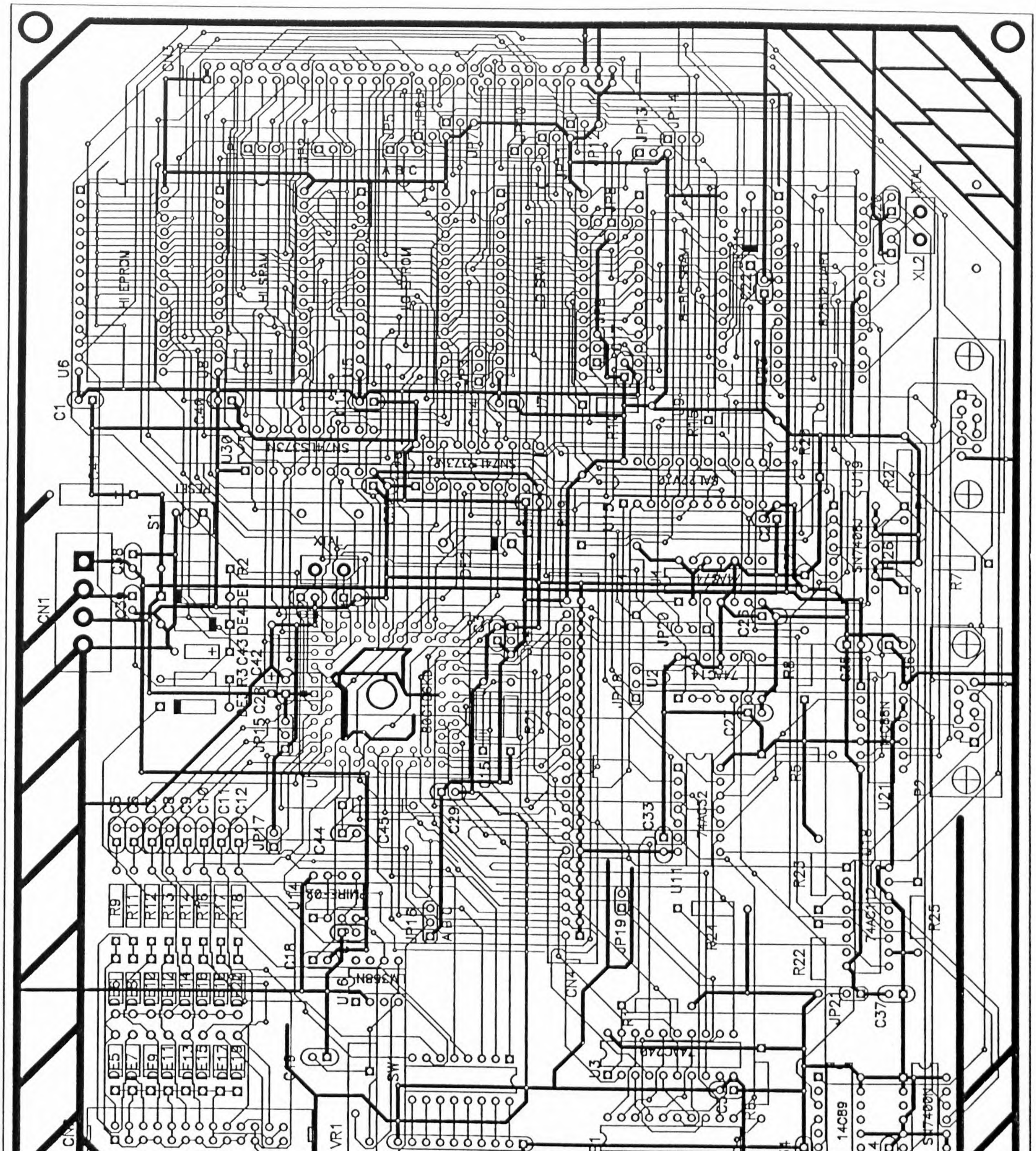

Transaction Download :
Tag Download Options
Download to Database
Download to file
Download to printer
View Latest Download File
Download File1 :
dnldf1.bin
Download File2 :
dnldf2.bin
Download File :
Download STATUS :
Cancel
Help
OK

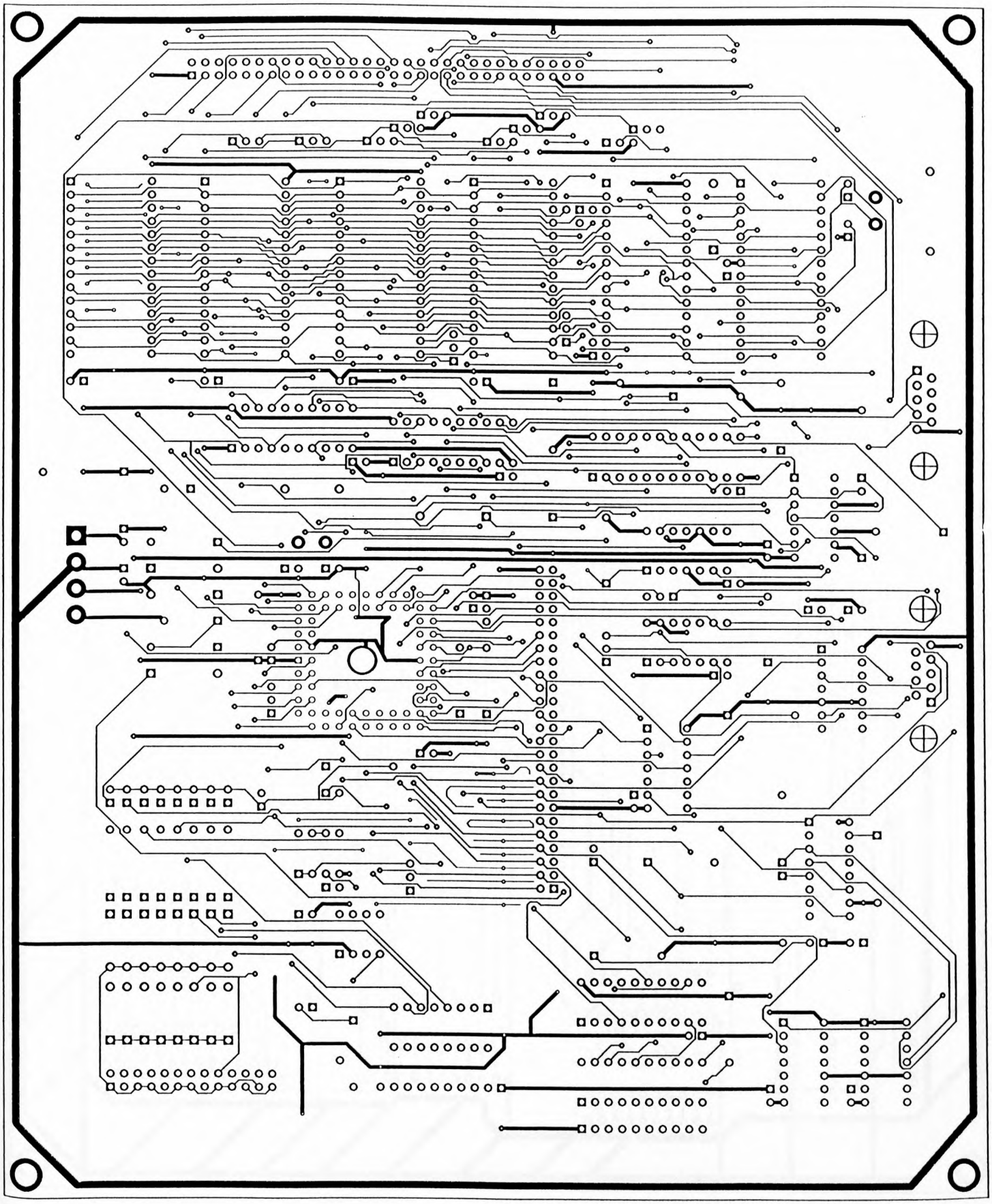
Appendix L

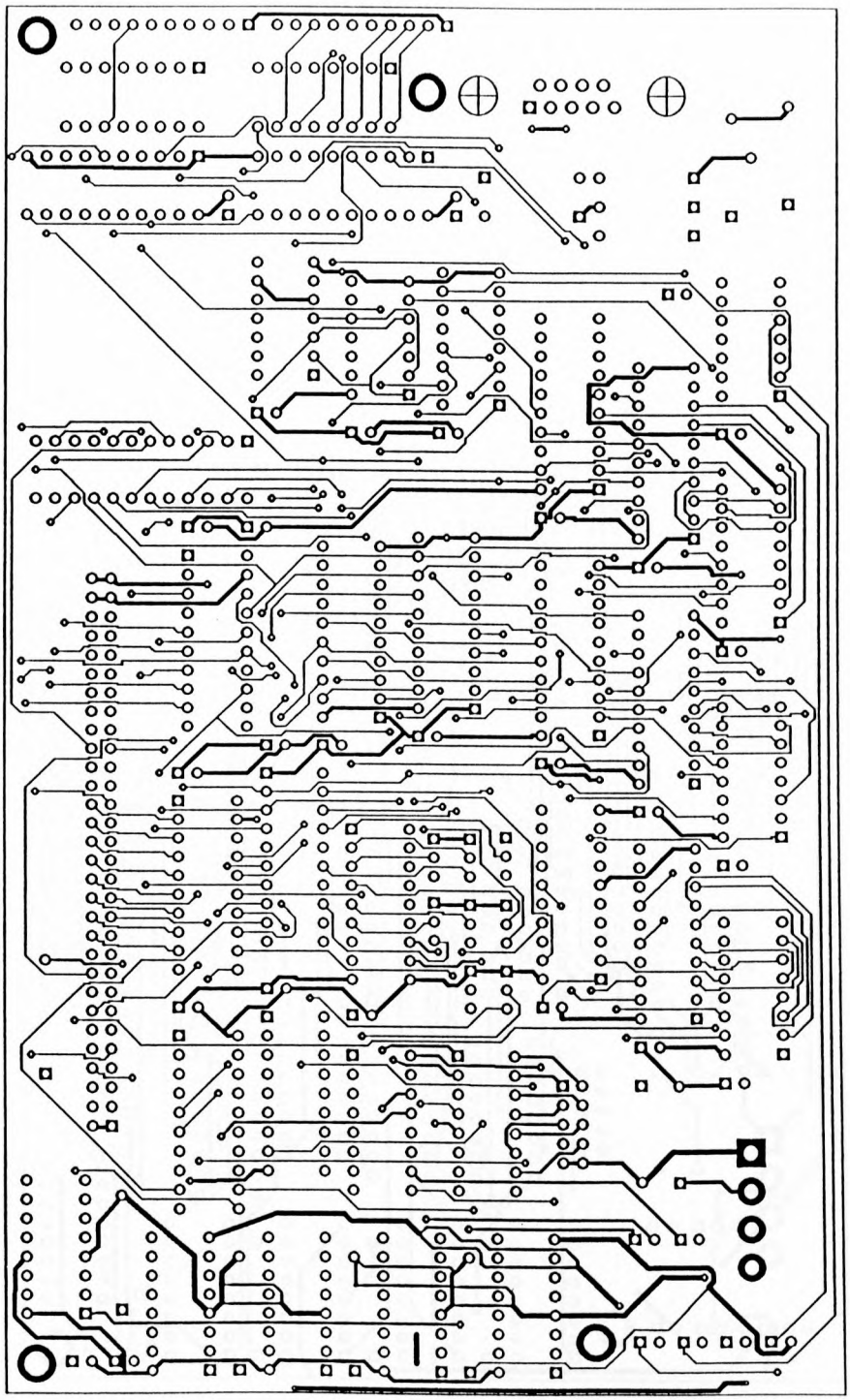
PHASE 1 Design Schematics

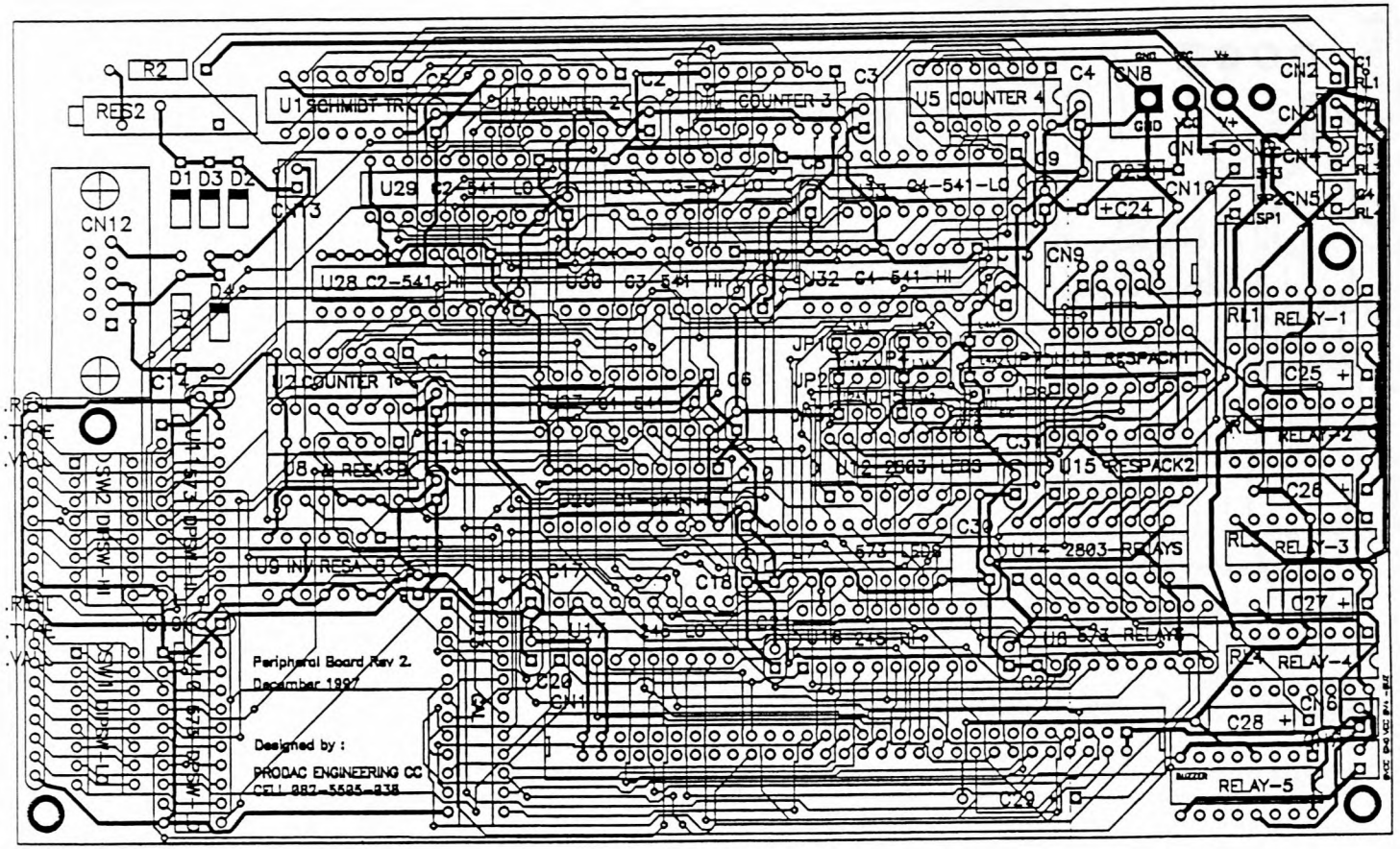
Appendix M

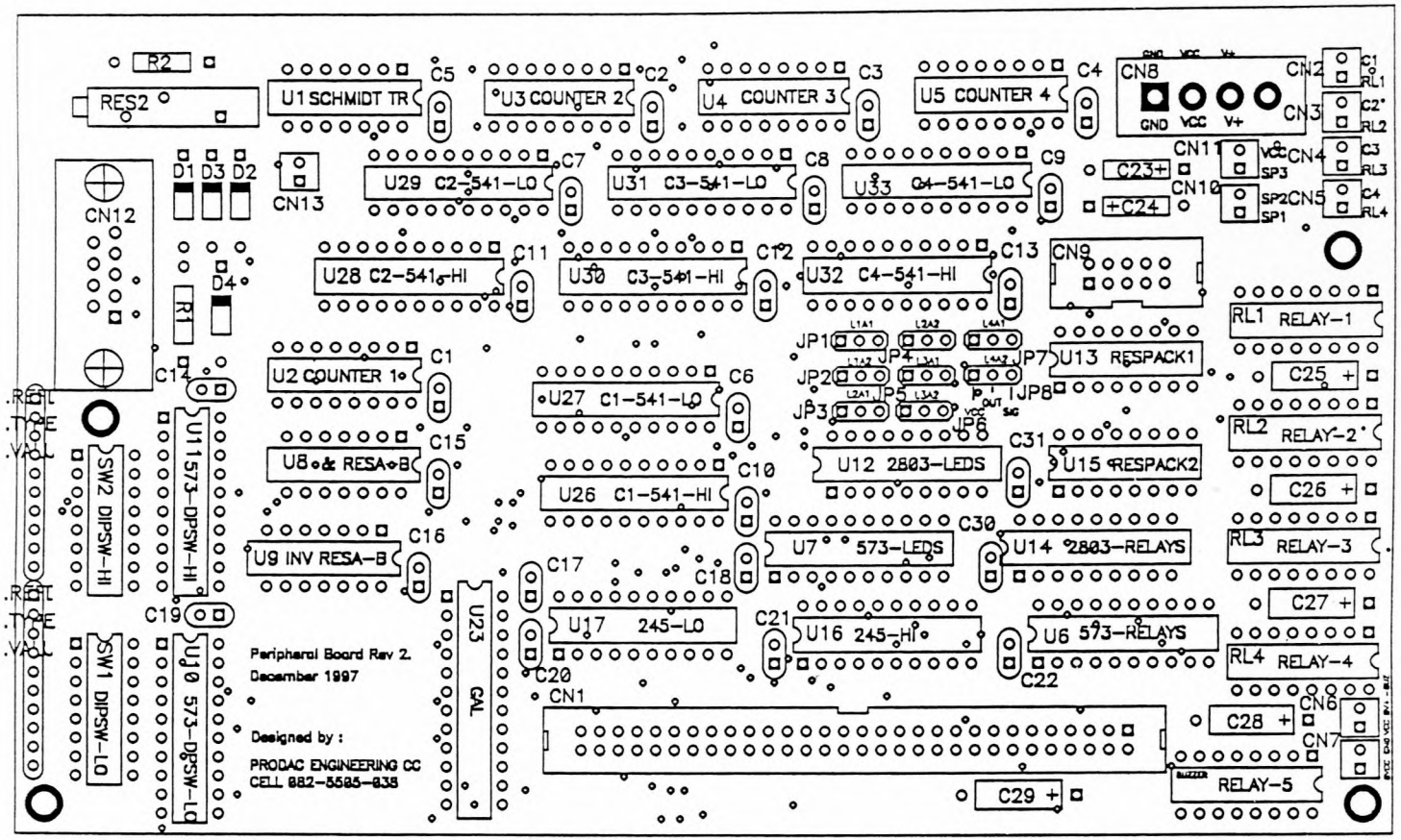
PHASE 1 Design PCBs


196 MICROCO TROLLER RE . 1 Jan. 1 97
Designe by: STEVEN S ANEPOEL
Call: 82-5505-0
HI EPROM
HI SRAM
LO EPROM
LO SRAM
SN74LS373N
GAL22V10
74AC14
74AC32
74AC240
74AC112
14C89
SN7400N
SN7405J
LM358N
XTAL
XL2
VR1
CN1
CN4
RESET



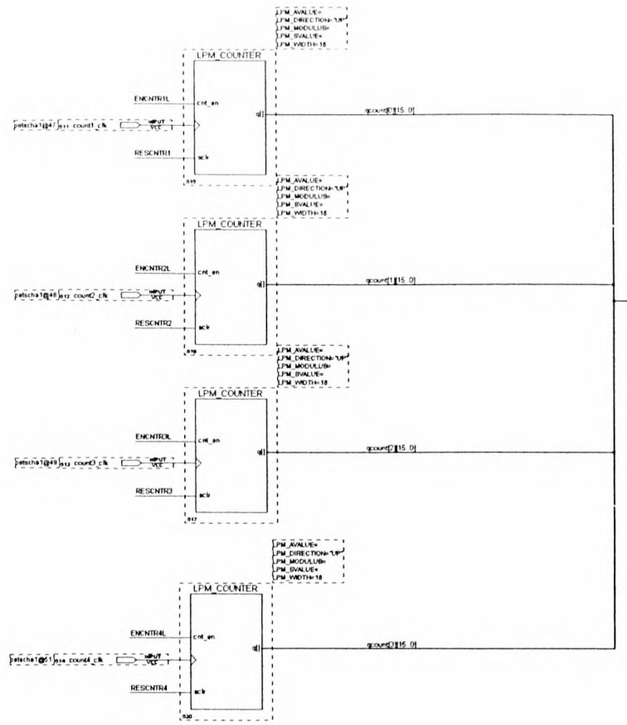


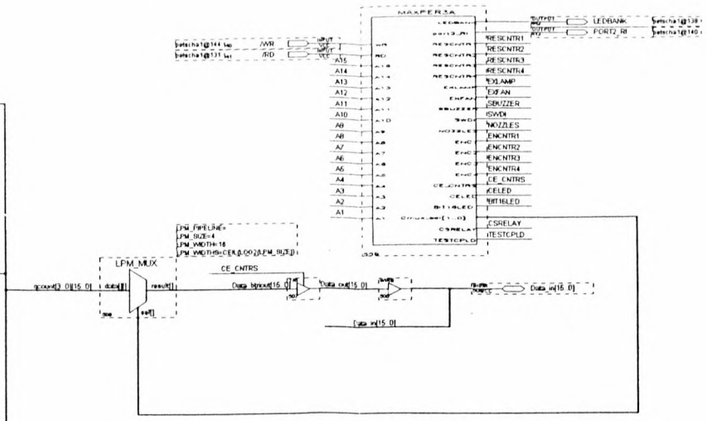

R2
RES2
U1 SCHMIDT TRG
U2 COUNTER 1
U3 COUNTER 2
COUNTER 3
U5 COUNTER 4
CN12
D1 D3 D2
D4
C13
U29 C2-541
U28 C2-541
U30 C3-541
U31 C4-541
U32 C4-541
U8 RESA4
U9 INV/RESA
U15 RESPACK2
U14 2803 RELAYS
RESPACK1
CN8
CN2
CN3
CN4
CN5
CN10
CN9
+C24
RL1 RELAY-1
C25 +
RELAY-2
C26 +
RELAY-3
C27 +
RELAY-4
C28 +
RELAY-5
SW2 DIP-SW
SW1 DIP-SW
Peripheral Board Rev 2.
December 1997
Designed by :
PRODAC ENGINEERING CC
CELL 082-5585-838

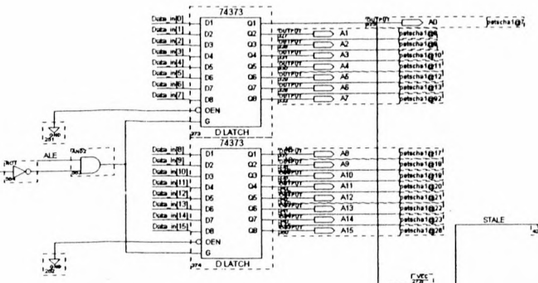

R2
RES2
U1 SCHMIDT TR
C5
U3 COUNTER 2
C2
U4 COUNTER 3
C3
U5 COUNTER 4
C4
CN8
GND VCC V+
CN2
CN3
C1
RL1
C2
RL2
C3
RL3
C4
RL4
CN11
VCC
SP3
CN4
CN10
SP2
SP1
CN5
C23+
+C24
D1 D3 D2
CN13
U29 C2-541-LO
C7
U31 C3-541-LO
C8
U33 C4-541-LO
C9
CN12
D4
R1
U28 C2-541-HI
C11
U30 C3-541-HI
C12
U32 C4-541-HI
C13
CN9
RL1 RELAY-1
C14
U2 COUNTER 1
C1
JP1
JP4
JP7
U13 RESPACK1
C25 +
U27 C1-541-LO
C6
JP2
JP5
JP8
RL2 RELAY-2
C15
U8 & RESA-B
JP3
JP6
C31
U15 RESPACK2
C26 +
U26 C1-541-HI
C10
U12 2803-LEDS
U11 573-DPSW-HI
SW2 DIPSW-HI
C16
U9 INV RESA-B
C30
U7 573-LEDS
U14 2803-RELAYS
RL3 RELAY-3
C17
C18
C27 +
C19
U17 245-LO
C21
U16 245-HI
U6 573-RELAYS
RL4 RELAY-4
SW1 DIPSW-LO
U10 573-DPSW-LO
Peripheral Board Rev 2.
December 1997
U23 CAL
C20
C22
CN1
C28 +
CN6
CN7
Designed by :
PRODAC ENGINEERING CC
CELL 082-5585-838
BUZZER
RELAY-5
C29 +

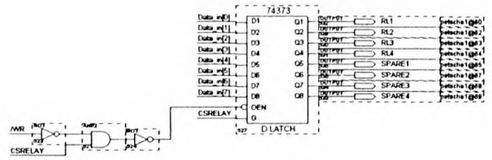
Appendix N

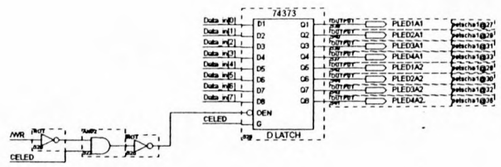
PHASE 2 Design's Graphics Definition Files

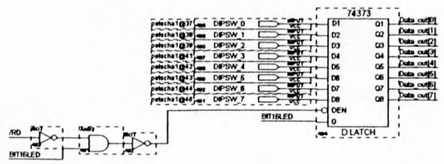

LPM_COUNTER
ENCNTR1L
RESCNTR1
LPM_COUNTER
ENCNTR2L
RESCNTR2
LPM_COUNTER
ENCNTR3L
RESCNTR3
LPM_COUNTER
ENCNTR4L
RESCNTR4

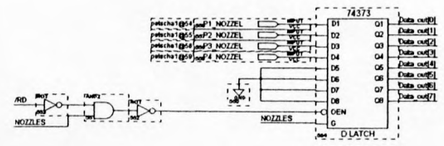

LPM_MUX
CE_CNTRS
MAPPER3A
RESCNTR1
RESCNTR2
RESCNTR3
RESCNTR4
EXLAMP
EXFAN
BUZZER
SWDI
NOZZLES
ENCNTR1
ENCNTR2
ENCNTR3
ENCNTR4
CE_CNTRS
CELED
WRTHLED
CSRELAY
ITESTCPLD

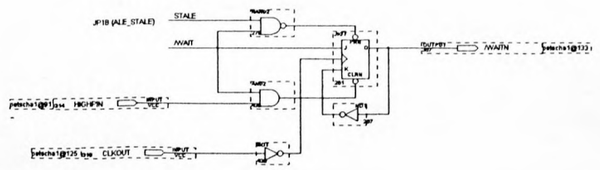

74373
D LATCH
ALE
CLKOUT
STALE
MAINCAL1
RESET
INST

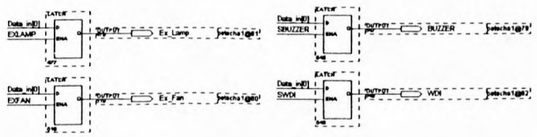

74373
D LATCH
CSRELAY
RL1
RL2
RL3
RL4
SPARE1
SPARE2
SPARE3
SPARE4

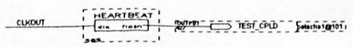

74373
D LATCH
CELED
PLED1A1
PLED2A1
PLED3A1
PLED4A1
PLED1A2
PLED2A2
PLED3A2
PLED4A2

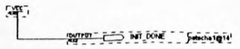

74373
D LATCH
DIPSW_0
DIPSW_1
DIPSW_2
DIPSW_3
DIPSW_4
DIPSW_5
DIPSW_6
DIPSW_7
BITBLED

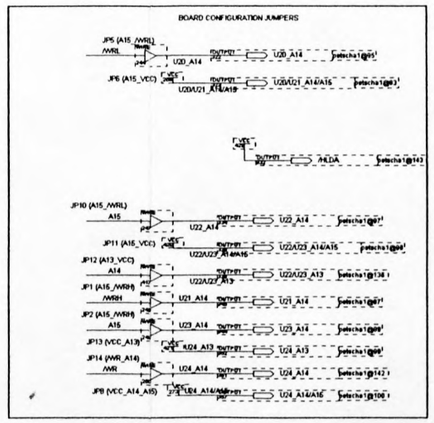

74373
D LATCH
P1_NOZZEL
P2_NOZZEL
P3_NOZZEL
P4_NOZZEL
NOZZLES

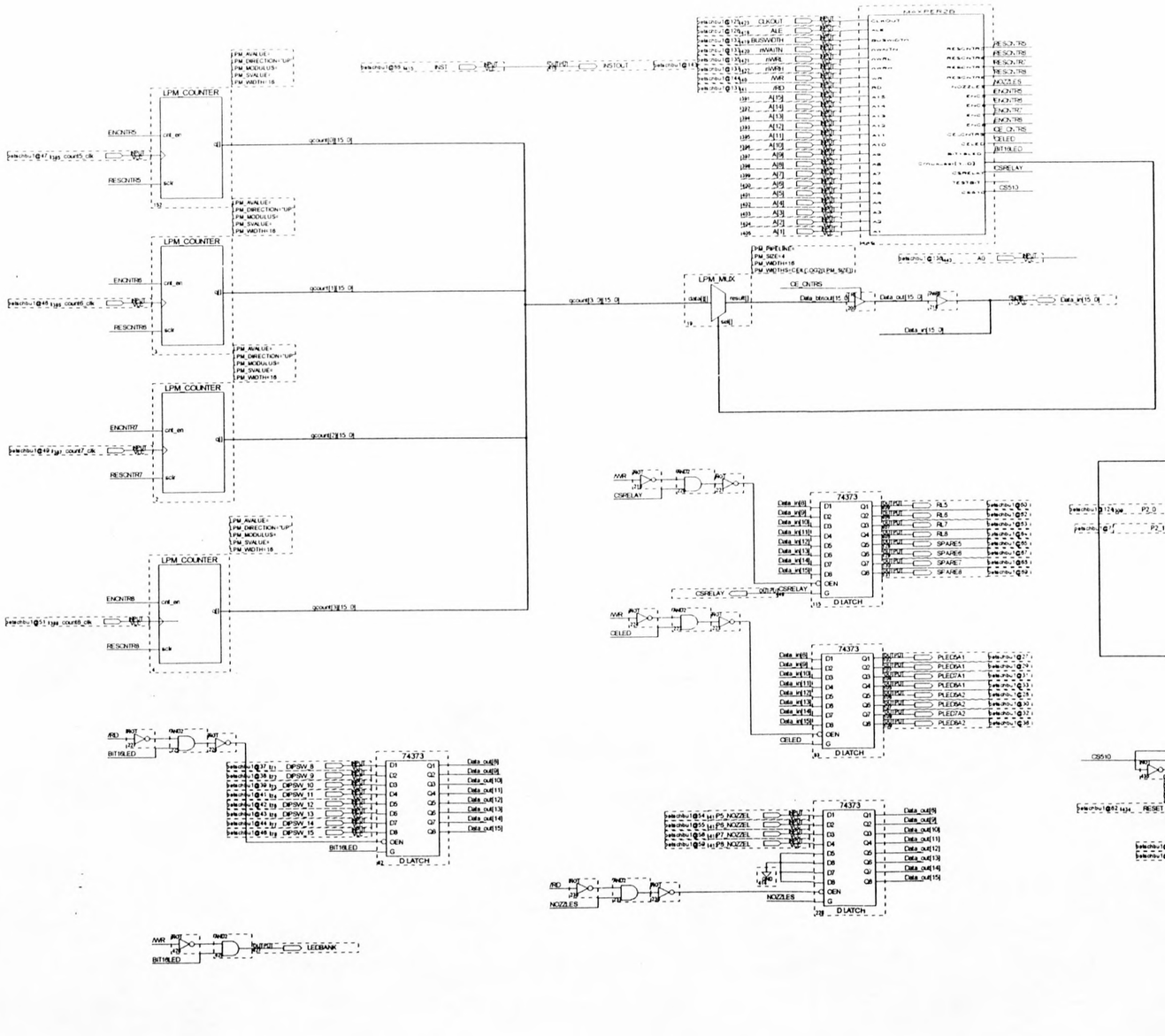

JP1B (ALE_STALE)
STALE
/WAIT
/WAITN
HIGHEN
CLKOUT


EXLAMP
Ex_Lamp
BUZZER
EXFAN
Ex_Fan
SWDI
WDI

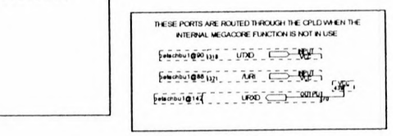

CLKOUT
HEARTBEAT
TEST_CPLD


INIT_DONE

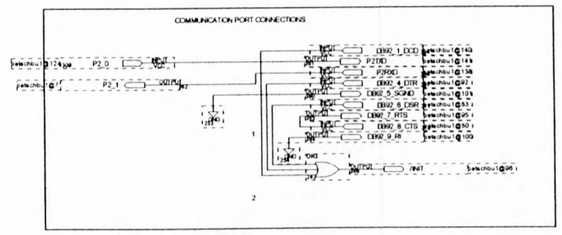

BOARD CONFIGURATION JUMPERS
JP5 (A15_/WRL)
JP6 (A15_VCC)
JP10 (A15_/WRL)
JP11 (A15_VCC)
JP12 (A13_VCC)
JP1 (A15_/WRH)
JP2 (A15_/WRH)
JP13 (VCC_A13)
JP14 (/WR_A14)
JP8 (VCC_A14_A15)

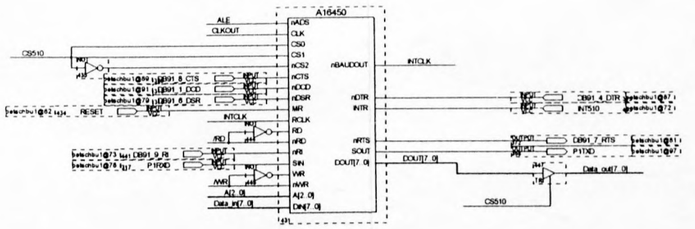


THESE PORTS ARE ROUTED THROUGH THE CPLD WHEN THE INTERNAL MEGACORE FUNCTION IS NOT IN USE



COMMUNICATION PORT CONNECTIONS





Appendix O

PHASE 2 Design's Text Definition Files

```
library IEEE;
    use IEEE.std_logic_1164.all;

entity maxper3A is
    port(WR          : in bit;
         RD          : in bit;
         A15         : in bit;
         A14         : in bit;
         A13         : in bit;
         A12         : in bit;
         A11         : in bit;
         A10         : in bit;
         A9          : in bit;
         A8          : in bit;
         A7          : in bit;
         A6          : in bit;
         A5          : in bit;
         A4          : in bit;
         A3          : in bit;
         A2          : in bit;
         A1          : in bit;
--       A0          : in bit;
         LEDBANK          : out bit;
         port2_RI         : out bit;
         RESCNTR1         : out bit;
         RESCNTR2         : out bit;
         RESCNTR3         : out bit;
         RESCNTR4         : out bit;
--       RESCNTR5         : out bit;
--       RESCNTR6         : out bit;
--       RESCNTR7         : out bit;
--       RESCNTR8         : out bit;
         EXLAMP         : out bit;
         EXFAN          : out bit;
         SBUZZER        :out bit;
         SWDI           :out bit;
         NOZZLES          : out bit;
         ENC1           : out bit;
         ENC2           : out bit;
         ENC3           : out bit;
         ENC4           : out bit;
--       ENC5           : out bit;
--       ENC6           : out bit;
--       ENC7           : out bit;
--       ENC8           : out bit;
         CE_CNTRS       : out bit;
         CELED        : out bit;
         BIT16LED     : out bit;
         Cmux_sel     : out std_logic_vector(1 downto 0);
         CSRELAY      : out bit;
         TESTCPLD     : out bit);
end maxper3A;

architecture FSM of maxper3A is
signal s1,CE1,CE2,CE3,CE4: bit;

begin
         s1         <= ((NOT A15) and (NOT A14) and (NOT A13) and A12 and A11 and A10 and
(NOT A9)and (NOT A8)and (NOT A7)and (NOT A6));
--       CE8        <= ((s1 and A5 and A4 and (NOT A3) and (NOT A2) and (NOT A1)) and
(NOT RD));
--       CE7        <= ((s1 and A5 and (NOT A4) and A3 and A2 and A1) and (NOT RD));
--       CE6        <= ((s1 and A5 and (NOT A4) and A3 and A2 and (NOT A1)) and (NOT
RD));
--       CE5        <= ((s1 and A5 and (NOT A4)and  A3 and (NOT A2) and A1) and (NOT
RD));
         CE4        <= ((s1 and (NOT A5) and (NOT A4)and (NOT A3) and (NOT A2) and (NOT
A1)) and (NOT RD)); --(ADDR: 1C00H)
         CE3        <= ((s1 and (NOT A5) and (NOT A4)and (NOT A3) and (NOT A2) and A1)
and (NOT RD));        --(ADDR: 1C02H)
```

```
        CE2       <= ((s1 and (NOT A5) and (NOT A4)and (NOT A3) and A2 and (NOT A1))
and (NOT RD));       --(ADDR: 1C04H)
        CE1       <= ((s1 and (NOT A5) and (NOT A4)and (NOT A3) and A2 and A1) and (NOT
RD));             --(ADDR: 1C06H)
--      RESCNTR8  <= ((NOT WR) and ((s1 and (NOT A5) and A4 and (NOT A3) and A2 and
A1)));
--      RESCNTR7  <= ((NOT WR) and ((s1 and (NOT A5) and A4 and (NOT A3) and A2 and
(NOT A1))));
--      RESCNTR6  <= ((NOT WR) and ((s1 and (NOT A5) and A4 and (NOT A3) and (NOT A2)
and A1)));
--      RESCNTR5  <= ((NOT WR) and ((s1 and (NOT A5) and A4 and (NOT A3) and (NOT A2)
and (NOT A1))));
        RESCNTR4  <= ((NOT WR) and ((s1 and (NOT A5) and (NOT A4) and A3 and A2 and
A1)));              --(ADDR: 1C0EH)
        RESCNTR3  <= ((NOT WR) and ((s1 and (NOT A5) and (NOT A4) and A3 and A2 and
(NOT A1))));        --(ADDR: 1C0CH)
        RESCNTR2  <= ((NOT WR) and ((s1 and (NOT A5) and (NOT A4) and A3 and (NOT A2)
and A1)));          --(ADDR: 1C0AH)
        RESCNTR1  <= ((NOT WR) and ((s1 and (NOT A5) and (NOT A4) and A3 and (NOT A2)
and (NOT A1))));    --(ADDR: 1C08H)
--      ENC8  <= ((NOT WR) and ((s1 and A5 and (NOT A4) and A3 and (NOT A2) and (NOT
A1))));
--      ENC7  <= ((NOT WR) and ((s1 and A5 and (NOT A4) and (NOT A3) and A2 and A1)));
--      ENC6  <= ((NOT WR) and ((s1 and A5 and (NOT A4) and (NOT A3) and A2 and (NOT
A1))));
--      ENC5  <= ((NOT WR) and ((s1 and A5 and (NOT A4) and (NOT A3) and (NOT A2) and
 1)));
        ENC4  <= ((NOT WR) and ((s1 and A5 and (NOT A4) and (NOT A3) and (NOT A2) and
(NOT A1))));        --(ADDR: 1C20H)
        ENC3  <= ((NOT WR) and ((s1 and (NOT A5) and A4 and A3 and A2 and A1)));
                --(ADDR: 1C1EH)
        ENC2  <= ((NOT WR) and ((s1 and (NOT A5) and A4 and A3 and A2 and (NOT A1))));
                --(ADDR: 1C1CH)
        ENC1  <= ((NOT WR) and ((s1 and (NOT A5) and A4 and A3 and (NOT A2) and A1)));
                --(ADDR: 1C1AH)
        BIT16LED  <= ((NOT RD) and ((s1 and (NOT A5) and (NOT A4) and (NOT A3) and A2
and (NOT A1))));    --(ADDR: 1C04H)
        CELED <= ((NOT WR) and ((s1 and (NOT A5) and (NOT A4) and (NOT A3) and A2 and
A1)));              --(ADDR: 1C06H)
        CSRELAY   <= ((NOT WR) and ((s1 and (NOT A5) and (NOT A4) and (NOT A3) and (NOT
A2) and (NOT A1))));--(ADDR: 1C00H)
        EXLAMP    <= ((NOT WR) and ((s1 and (NOT A5) and A4 and A3 and (NOT A2) and
(NOT A1))));         --(ADDR: 1C18H)
        EXFAN     <= ((NOT WR) and ((s1 and A5 and A4 and (NOT A3) and (NOT A2) and
A1)));               --(ADDR: 1C32H)
        SBUZZER   <= ((NOT WR) and ((s1 and A5 and A4 and (NOT A3) and A2 and A1)));
                --(ADDR: 1C02H)*
        SWDI  <= ((NOT WR) and ((s1 and A5 and A4 and A3 and (NOT A2) and (NOT A1))));
                --(ADDR: 1C38H)
        NOZZLES   <= ((NOT RD) and ((s1 and A5 and A4 and (NOT A3) and A2 and (NOT
A1))));              --(ADDR: 1C34H)
        CE_CNTRS <= CE1 OR CE2 OR CE3 OR CE4;
        LEDBANK   <= ((NOT WR) and ((s1 and A5 and A4 and A3 and (NOT A2) and A1)));
                --(ADDR: 1C3AH)
        port2_RI  <= ((NOT WR) and ((s1 and A5 and A4 and A3 and A2 and (NOT A1))));
                --(ADDR: 1C3CH)
        TESTCPLD  <= ((NOT WR) and ((s1 and A5 and A4 and A3 and A2 and A1)));
                --(ADDR: 1C3EH)
--      Cmux_sel <= "00";

process

begin
        --  CE_CNTRS <= '1';        -- none of the counter outputs are latched onto the
output bus
    IF CE1 = '1' THEN
        Cmux_sel <= "00";
--      CE_CNTRS <= '0';
    ELSIF CE2 = '1' THEN
        Cmux_sel <= "01";
--      CE_CNTRS <= '0';
    ELSIF CE3 = '1' THEN
```

```
        Cmux_sel <= "10";
--      CE_CNTRS <= '0';
    ELSIF CE4 = '1' THEN
        Cmux_sel <= "11";
--      CE_CNTRS <= '0';
    END IF;

end process;
end FSM;
```

```
-- Steven Swanepoel's VHDL Template for a heartbeat indicator

library IEEE;
    use IEEE.std_logic_1164.all;

ENTITY heartbeat IS

    PORT
    (
        clk         : IN    STD_LOGIC;  --  clock used to increment the counter
        flash       : OUT   STD_LOGIC   --  output from heartbeat , indicates a time for
toggle of the heartbeat
    );

END heartbeat;

ARCHITECTURE vhdl_code OF heartbeat IS
    signal toggle,time_out : STD_LOGIC;
BEGIN

    PROCESS (clk) -- 1 clock cycle = 100ns (10MHz)
    VARIABLE count : integer range 0 to 2500000;
    BEGIN

        IF (clk'EVENT and clk = '1') THEN   -- enter this code section on pos clock
transistion only
                IF count <= 2500000 THEN    -- if counter <= 1000 , continue counting
                    count := count + 1;
                    time_out<='0';
                ELSE
                    time_out<='1';  -- produce a counter lapse signal , used for e.g. cpu
reset
                    IF toggle = '1' THEN
                        toggle <= '0';
                    ELSE
                        toggle <= '1';
                    END IF;
                END IF;
        END IF;
    IF time_out = '1' THEN
            count := 0;       -- reset counter
            IF toggle = '0' THEN
                FLASH <= '0';
            ELSE
                FLASH <= '1';
            END IF;
    END IF;
    END PROCESS;

END vhdl_code;
```

```
library IEEE;
    use IEEE.std_logic_1164.all;

entity maingal1 is
PORT(
    clockout: in std_logic;
    stale: in std_logic;
    holda: in std_logic;
    a8: in std_logic;
    a9: in std_logic;
    a10: in std_logic;
    a11: in std_logic;
    a12: in std_logic;
    a13: in std_logic;
    a14: in std_logic;
    a15: in std_logic;
    reset: in std_logic;
    cs510: out std_logic;
    ce2: out std_logic;
    buswidth: out std_logic;
    out0: out std_logic;
    waite: out std_logic;
    ce0: out std_logic;
    ce1: out std_logic;
    inst: in std_logic;
    LEDBANK: in std_logic
--  out1: out std_logic
    );

end maingal1;

architecture fsm of maingal1 is
    signal s0, s1, s2, s3, s4, s5, s6, s7, s8, s9, s10, s11, s12, s13, s14, s15, s16,
s17, s18, s19, s20, s21, s22: std_logic;
    signal nwait_1, nwait_2, nwait_3, nwait_4, mappd1, msig, msig1, waitt : std_logic;
    signal state:std_logic_vector (1 downto 0);
    constant async_start: std_logic_vector (1 downto 0) := "00";
    constant hold_2: std_logic_vector (1 downto 0) := "01";
    constant hold_3: std_logic_vector (1 downto 0) := "10";
    constant remove_hold: std_logic_vector (1 downto 0) := "11";

begin
    s0  <= (NOT A15) and (NOT A14) and (NOT A13) and (NOT A12);  -- 0000H - 0FFFH
    s1  <= (NOT A15) and (NOT A14) and A13 and (NOT A12);  -- 2000H
    s2  <= (NOT A15) and (NOT A14) and A13 and A12;        -- 3000H
    s3  <= (NOT A15) and A14 and (NOT A13) and (NOT A12);  -- 4000H
    s4  <= (NOT A15) and A14 and (NOT A13) and A12;        -- 5000H
    s5  <= (NOT A15) and A14 and A13 and (NOT A12);        -- 6000H
    s6  <= (NOT A15) and A14 and A13 and A12;              -- 7000H
    s7  <= A15 and (NOT A14) and (NOT A13) and (NOT A12);  -- 8000H
    s8  <= A15 and (NOT A14) and (NOT A13) and A12;        -- 9000H
    s9  <= A15 and (NOT A14) and A13 and (NOT A12);        -- A000H
    s10 <= A15 and (NOT A14) and A13 and A12;           -- B000H
    s11 <= A15 and A14 and (NOT A13) and (NOT A12);     -- C000H
    s12 <= A15 and A14 and (NOT A13) and A12;           -- D000H
    s13 <= A15 and A14 and A13 and (NOT A12);           -- E000H
    s14     <= (A15 and A14 and A13 and A12);             -- F000H
    s15     <= (s0 or s21);
    s16     <= (s1 or s2 or s3 or s4 or s5 or s6 or s7 or s8 or s9 or s10);  -- 2000H -
BFFFH 16 bit eprom sim
    s17     <= (NOT A15) and (NOT A14) and (NOT A13) and A12 and A11 and A10 and A9 and
(NOT A8); --1E00H -1EFFH UART
    s18     <= (s19 or s17);  -- eprom sim or UART (buswidth)
    s19     <= (s11 or s12 or s13 or s14);  -- C000H - FFFFH 8 bit real time clock nvram
    s20     <= (NOT A15) and (NOT A14) and (NOT A13) and A12 and A11 and A10 and (NOT A9)
and (NOT A8); --1C00H - 1DFFH PERIPH
    s21   <= (NOT A15) and (NOT A14) and (NOT A13) and A12 and (NOT A11) and (NOT A10)
and (NOT A9); --1000H -11FFH 16 - bit RAM
```

```
    --nwait_1      <= '0';
    --nwait_2      <= '0';

    nwait_1     <= stale and  holda and (nwait_2 or s16 or s0); -- 8 bit eprom sim
--  nwait_1     <= (nwait_2 or s16); -- 8 bit eprom sim
--  1 wait state for eproms
    nwait_2     <= stale and  holda and (nwait_3 or s17 or s19 or s20 or LEDBANK);
                --NOT A8 SHOULD COME OUT
--  nwait_2     <= (nwait_3 or s17 or s19 or s20);                          --NOT A8 SHOULD
COME OUT
--  2 wait states for sram, nvram & peripheral outputs (for 4040 counter ics)
    nwait_3     <= '0';
    nwait_4     <= INST;
--  3 wait states not required
--   ce2       <= (NOT s19) AND (NOT inst);  -- 8 bit real time clock nvram

--  when the INST pin is high indicates instruction is fetched otherwise data is fetched
--          buswidth <= NOT s18;
            buswidth <= NOT s18;
            ce0      <= NOT s16;  -- 16 bit sim or EPROMS (ADDR: 2000H - BFFFH)
            ce2      <= (NOT s19); --AND (NOT inst);  -- 8 bit real time clock nvram
(ADDR: C000H - FFFFH)
            ce1      <= NOT s15; --AND (NOT inst);  -- 16 bit sram slots for stack (ADDR:
0000H - 0FFFH and 1000H -11FFH)
            cs510    <= NOT s17;  -- EXTERNAL UART (ADDR: 1E00H -1EFFH)

 -   ce1      <= NOT s15;  -- 16 bit sram slots for stack
--   cs510   <= NOT s17;  -- UART
--   buswidth <= NOT s18;
--   ce0         <= NOT s16;  -- 16 bit sim
     waite <= not waitt;

process(reset, clockout)
begin
        if (reset = '0') then
--          buswidth <= '1';
            state <= async_start;
--          ce0          <= '1';
--          ce2      <= '1';
--          ce1      <= '1';
--          cs510    <= '1';
        elsif (clockout'event and clockout = '1') then
--          buswidth <= NOT s18;
--          ce0      <= NOT s16;  -- 16 bit sim or EPROMS (ADDR: 2000H - BFFFH)
--          ce2      <= (NOT s19); --AND (NOT inst);  -- 8 bit real time clock nvram
(ADDR: C000H - FFFFH)
--          ce1      <= NOT s15; --AND (NOT inst);  -- 16 bit sram slots for stack (ADDR:
 0000H - 0FFFH and 1000H -11FFH)
--          cs510    <= NOT s17;  -- EXTERNAL UART (ADDR: 1E00H -1EFFH)
 --chip selects are placed here in order to guarentee a spike free
-- output since it is a clocked output
            case state is
                when async_start =>
                    if (nwait_1 = '1') then
                        waitt <= '1';
--                      waite <= '0';
                    end if;
                    if (nwait_1 = '1' and nwait_2 = '0') then
                        state <= remove_hold;
                    elsif (nwait_2 = '1') then
                        state <= hold_2;
                    else
                        state <= async_start;
                    end if;
                when hold_2 =>
                        waitt <= '1';
--                      waite <= '0';
                    if (nwait_3 = '1') then
                        state <= hold_3;
                    else
                        state <= remove_hold;
                    end if;
```

```
                when hold_3 =>
                        waitt <= '1';
--                      waite <= '0';
                        state <= remove_hold;
                when remove_hold =>
                        waitt <= '0';
--                      waite <= '1';
                        state <= async_start;
                when others =>
                    state <= async_start;
            end case;
        end if;
    end process;
out0 <= state(0);
--out1 <= state(1);
end fsm;
```

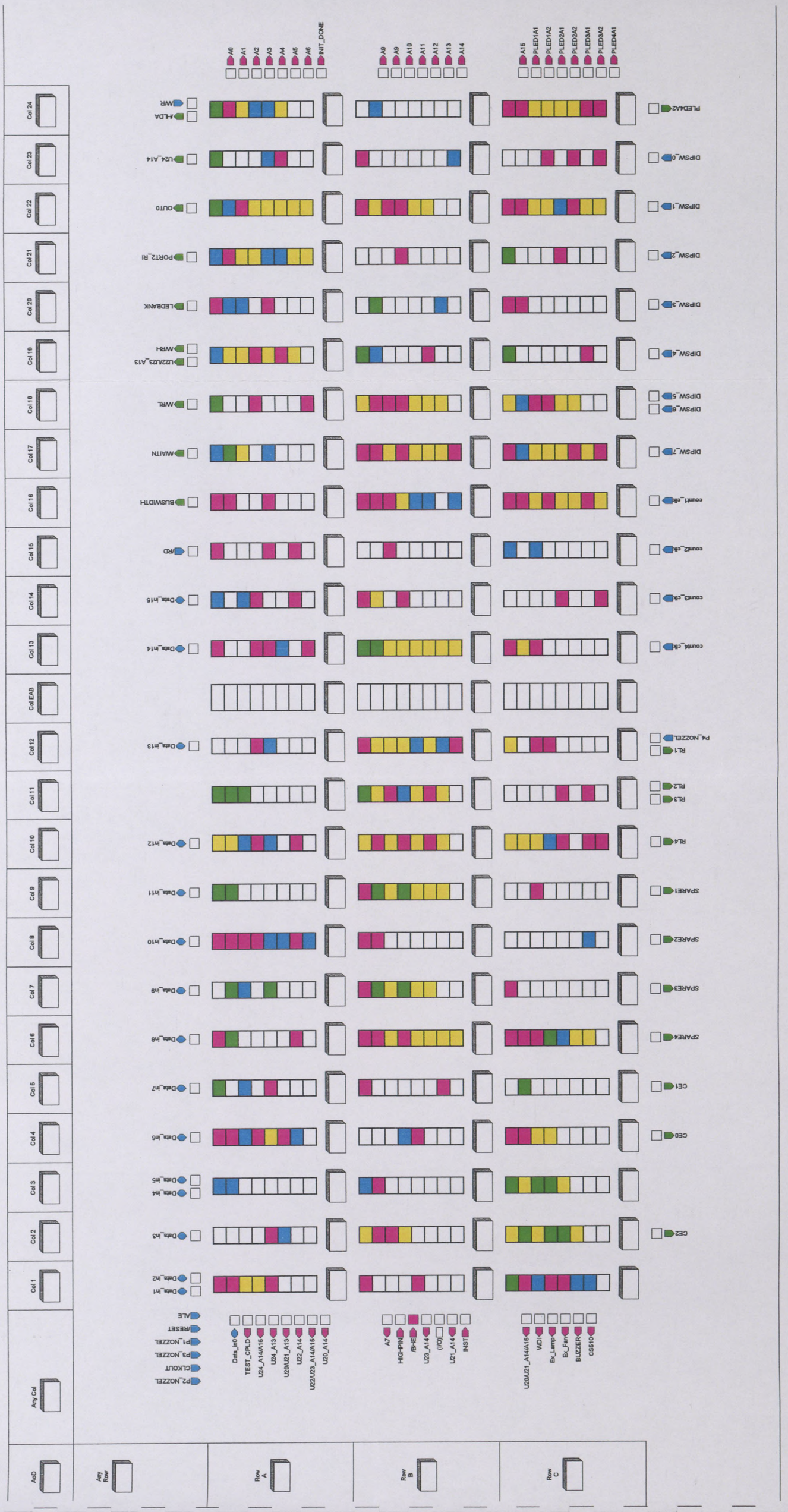
```
library IEEE;
   use IEEE.std_logic_1164.all;

entity maxper2B is
    port(CLKOUT          : in bit;
         ALE          : in bit;
         BUSWIDTH     : in bit;
         nWAITN       : in bit;
         nWRL         : in bit;
         nWRH         : in bit;
         WR           : in bit;
         RD           : in bit;
         A15          : in bit;
         A14          : in bit;
         A13          : in bit;
         A12          : in bit;
         A11          : in bit;
         A10          : in bit;
         A9           : in bit;
         A8           : in bit;
         A7           : in bit;
         A6           : in bit;
         A5           : in bit;
         A4           : in bit;
         A3           : in bit;
         A2           : in bit;
         A1           : in bit;
--       A0           : in bit;
--       RESCNTR1        : out bit;
--       RESCNTR2        : out bit;
--       RESCNTR3        : out bit;
--       RESCNTR4        : out bit;
         RESCNTR5        : out bit;
         RESCNTR6        : out bit;
         RESCNTR7        : out bit;
         RESCNTR8        : out bit;
--       EXLAMP        : out bit;
--       EXFAN         : out bit;
--       SBUZZER       :out bit;
--       SWDI          :out bit;
         NOZZLES         : out bit;
--       ENC1          : out bit;
--       ENC2          : out bit;
--       ENC3          : out bit;
--       ENC4          : out bit;
         ENC5          : out bit;
         ENC6          : out bit;
         ENC7          : out bit;
         ENC8          : out bit;
         CE_CNTRS      : out bit;
         CELED         : out bit;
         BIT16LED      : out bit;
         Cmux_sel      : out std_logic_vector(1 downto 0);
         CSRELAY       : out bit;
         TESTBIT       : out bit);
end maxper2B;

architecture FSM of maxper2B is
signal s1,CE5,CE6,CE7,CE8: bit;

begin
         s1         <= ((NOT A15) and (NOT A14) and (NOT A13) and A12 and A11 and A10 and
(NOT A9)and (NOT A8)and (NOT A7)and (NOT A6));
         CE8        <= ((s1 and A5 and A4 and (NOT A3) and (NOT A2) and (NOT A1)) and
(NOT RD));
         CE7        <= ((s1 and A5 and (NOT A4) and A3 and A2 and A1) and (NOT RD));
         CE6        <= ((s1 and A5 and (NOT A4) and A3 and A2 and (NOT A1)) and (NOT
RD));
         CE5        <= ((s1 and A5 and (NOT A4)and  A3 and (NOT A2) and A1) and (NOT
RD));
```

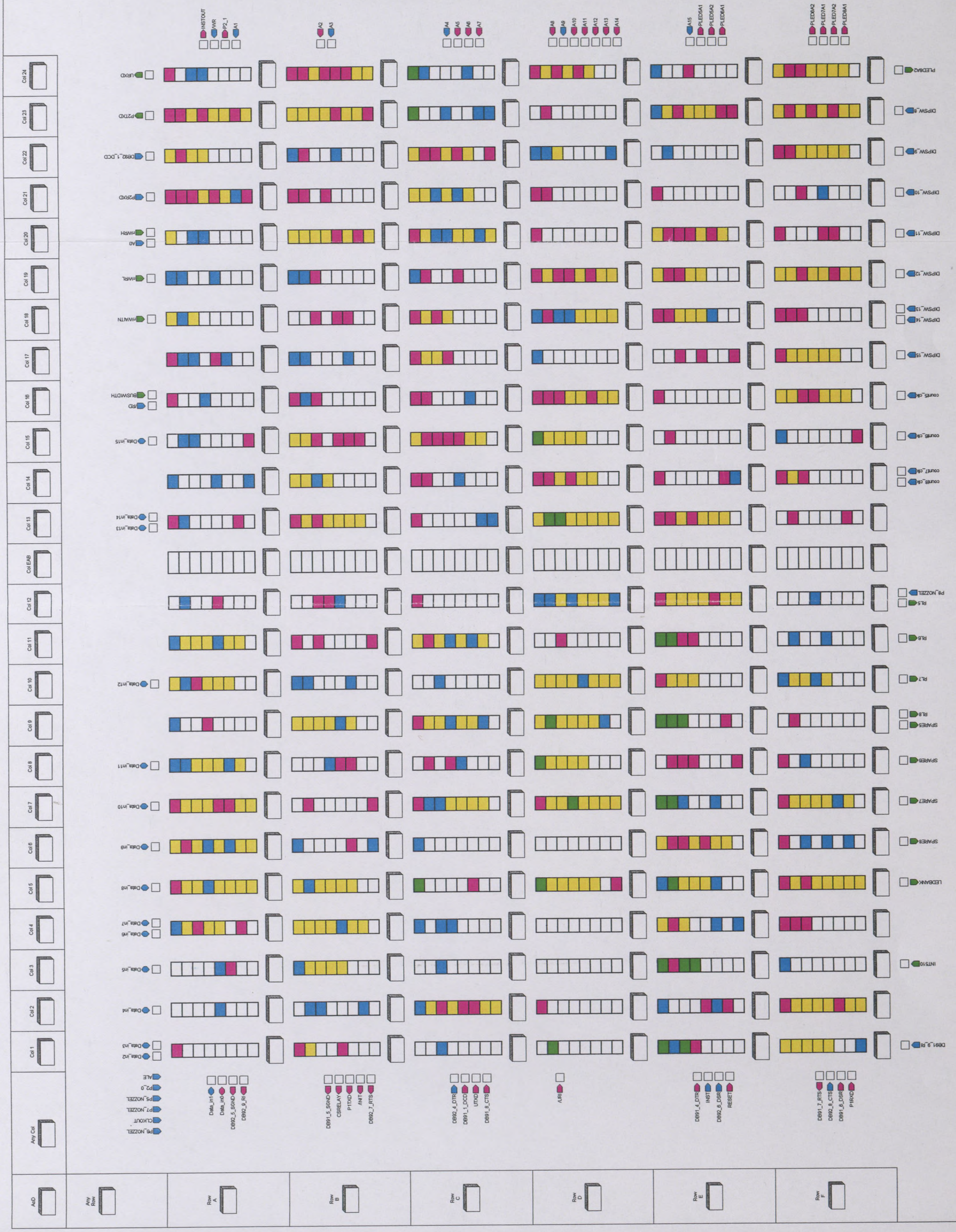
```
--        CE4       <= ((s1 and (NOT A5) and (NOT A4)and (NOT A3) and (NOT A2) and (NOT
A1)) and (NOT RD));
--        CE3       <= ((s1 and (NOT A5) and (NOT A4)and (NOT A3) and (NOT A2) and A1)
and (NOT RD));
--        CE2       <= ((s1 and (NOT A5) and (NOT A4)and (NOT A3) and A2 and (NOT A1))
and (NOT RD));
--        CE1       <= ((s1 and (NOT A5) and (NOT A4)and (NOT A3) and A2 and A1) and (NOT
RD));
          RESCNTR8  <= ((NOT WR) and ((s1 and (NOT A5) and A4 and (NOT A3) and A2 and
A1)));
          RESCNTR7  <= ((NOT WR) and ((s1 and (NOT A5) and A4 and (NOT A3) and A2 and
(NOT A1))));
          RESCNTR6  <= ((NOT WR) and ((s1 and (NOT A5) and A4 and (NOT A3) and (NOT A2)
and A1)));
          RESCNTR5  <= ((NOT WR) and ((s1 and (NOT A5) and A4 and (NOT A3) and (NOT A2)
and (NOT A1))));
--        RESCNTR4  <= ((NOT WR) and ((s1 and (NOT A5) and (NOT A4) and A3 and A2 and
A1)));
--        RESCNTR3  <= ((NOT WR) and ((s1 and (NOT A5) and (NOT A4) and A3 and A2 and
(NOT A1))));
--        RESCNTR2  <= ((NOT WR) and ((s1 and (NOT A5) and (NOT A4) and A3 and (NOT A2)
and A1)));
--        RESCNTR1  <= ((NOT WR) and ((s1 and (NOT A5) and (NOT A4) and A3 and (NOT A2)
and (NOT A1))));
          ENC8  <= ((NOT WR) and ((s1 and A5 and (NOT A4) and A3 and (NOT A2) and (NOT
A1))));
          ENC7  <= ((NOT WR) and ((s1 and A5 and (NOT A4) and (NOT A3) and A2 and A1)));
          ENC6  <= ((NOT WR) and ((s1 and A5 and (NOT A4) and (NOT A3) and A2 and (NOT
A1))));
          ENC5  <= ((NOT WR) and ((s1 and A5 and (NOT A4) and (NOT A3) and (NOT A2) and
A1)));
--        ENC4  <= ((NOT WR) and ((s1 and A5 and (NOT A4) and (NOT A3) and (NOT A2) and
(NOT A1))));
--        ENC3  <= ((NOT WR) and ((s1 and (NOT A5) and A4 and A3 and A2 and A1)));
--        ENC2  <= ((NOT WR) and ((s1 and (NOT A5) and A4 and A3 and A2 and (NOT A1))));
--        ENC1  <= ((NOT WR) and ((s1 and (NOT A5) and A4 and A3 and (NOT A2) and A1)));
          BIT16LED  <= ((NOT WR) and ((s1 and (NOT A5) and (NOT A4) and (NOT A3) and A2
and (NOT A1))));
          CELED <= ((NOT WR) and ((s1 and (NOT A5) and (NOT A4) and (NOT A3) and A2 and
A1)));
          CSRELAY   <= ((NOT WR) and ((s1 and (NOT A5) and (NOT A4) and (NOT A3) and (NOT
A2) and (NOT A1))));
--        EXLAMP    <= ((NOT WR) and ((s1 and (NOT A5) and A4 and A3 and (NOT A2) and
(NOT A1))));
--        EXFAN     <= ((NOT WR) and ((s1 and A5 and A4 and (NOT A3) and (NOT A2) and
A1)));
--        SBUZZER   <= ((NOT WR) and ((s1 and A5 and A4 and (NOT A3) and A2 and A1)));
--        SWDI  <= ((NOT WR) and ((s1 and A5 and A4 and A3 and (NOT A2) and (NOT A1))));
          NOZZLES   <= ((NOT WR) and ((s1 and A5 and A4 and (NOT A3) and A2 and (NOT
A1))));
          CE_CNTRS <= CE5 OR CE6 OR CE7 OR CE8;
--          Cmux_sel <= "000";
          TESTBIT <=  CLKOUT OR ALE OR BUSWIDTH OR nWAITN OR nWRL OR nWRH;

process

begin
       --  CE_CNTRS <= '1';        -- none of the counter outputs are latched onto the
output bus
    IF CE5 = '1' THEN
          Cmux_sel <= "00";
--        CE_CNTRS <= '0';
    ELSIF CE6 = '1' THEN
          Cmux_sel <= "01";
--        CE_CNTRS <= '0';
    ELSIF CE7 = '1' THEN
          Cmux_sel <= "10";
--        CE_CNTRS <= '0';
    ELSIF CE8 = '1' THEN
          Cmux_sel <= "11";
--        CE_CNTRS <= '0';
  --    ELSIF CE5 = '1' THEN
```

```
  --          Cmux_sel <= "100";
--          CE_CNTRS <= '0';
  --      ELSIF CE6 = '1' THEN
  --          Cmux_sel <= "101";
--          CE_CNTRS <= '0';
  --      ELSIF CE7 = '1' THEN
  --          Cmux_sel <= "110";
--          CE_CNTRS <= '0';
  --      ELSIF CE8 = '1' THEN
  --          Cmux_sel <= "111";
--          CE_CNTRS <= '0';
--    ELSE
--          CE_CNTRS <= '1';       -- none of the counter outputs are latched onto the
output bus

    END IF;

end process;


end FSM;
```

Appendix P

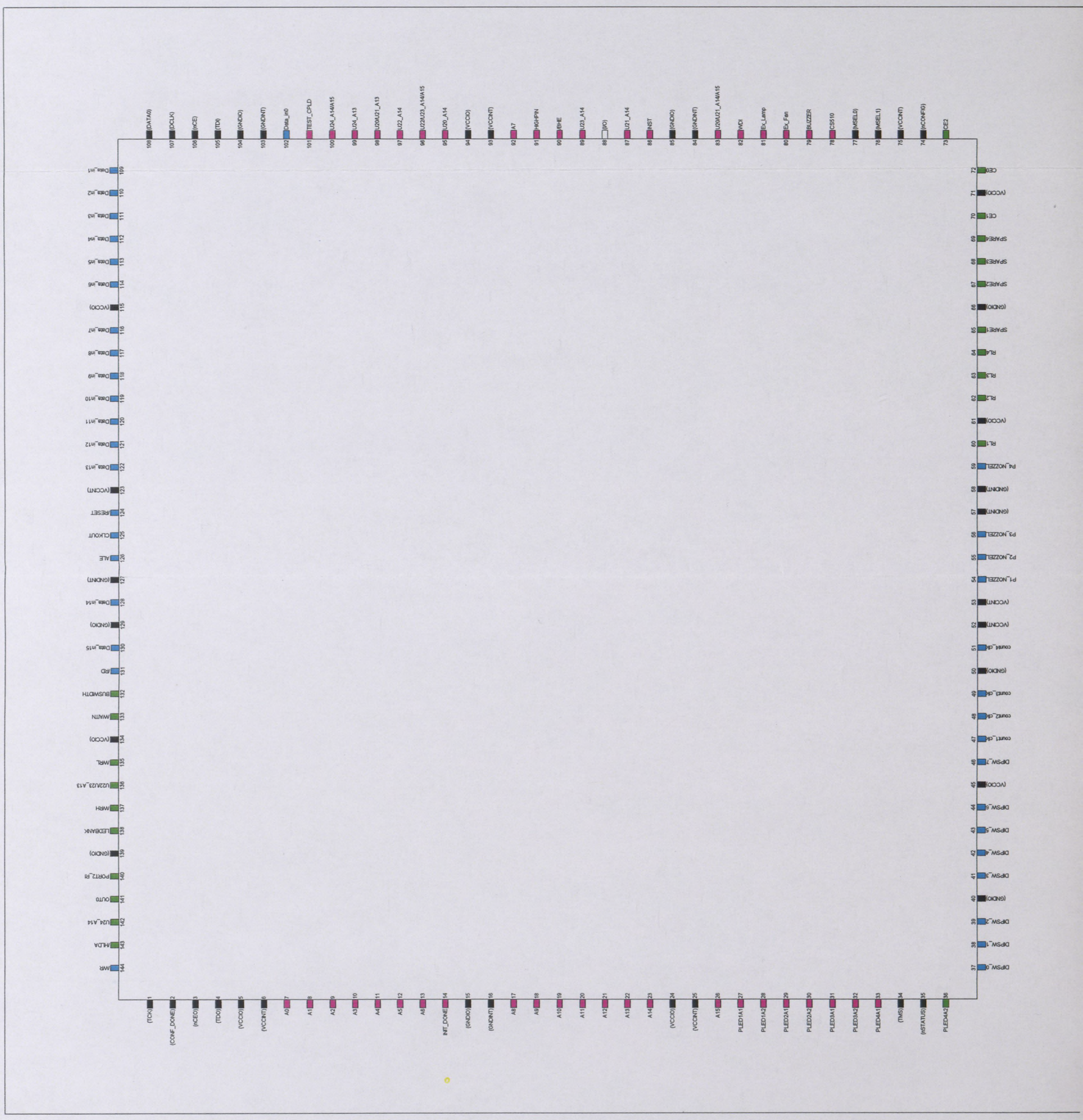
Synthesis of the FLEX 10Ks Internal Design

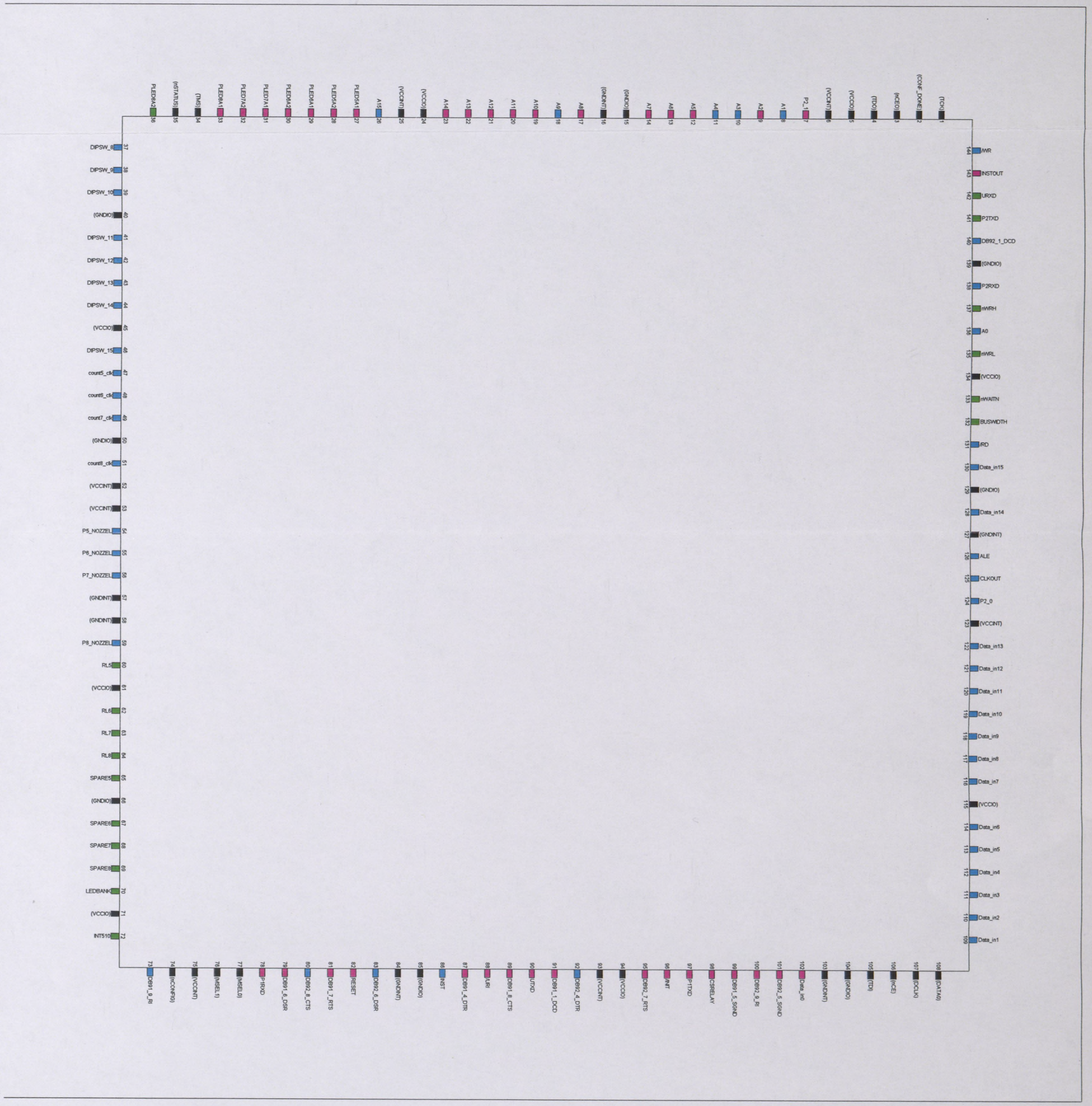

A0
A1
A2
A3
A4
A5
A6
INIT_DONE
A8
A9
A10
A11
A12
A13
A14
A15
PLED1A1
PLED1A2
PLED2A1
PLED2A2
PLED3A1
PLED3A2
PLED4A1
PLED4A2
DIPSW_0
DIPSW_1
DIPSW_2
DIPSW_3
DIPSW_4
DIPSW_5
DIPSW_6
DIPSW_7
count1_clk
count2_clk
count3_clk
count4_clk
RL1
RL2
RL3
RL4
SPARE1
SPARE2
SPARE3
SPARE4
CE1
CE0
CE2
P4_NOZZEL
Data_in13
Data_in14
Data_in15
RD
BUSWIDTH
MATN
MRL
U22/U23_A13
MRH
LEDBANK
PORT2_RI
OUT0
U24_A14
ALDA
MWR
Data_in1
Data_in2
Data_in3
Data_in4
Data_in5
Data_in6
Data_in7
Data_in8
Data_in9
Data_in10
Data_in11
Data_in12
Data_in13
P2_NOZZEL
CLKOUT
P3_NOZZEL
P1_NOZZEL
RESET
ALE
Data_in0
TEST_CPLD
U24_A14/A15
U24_A13
U20/U21_A13
U22_A14
U22/U23_A14/A15
U20_A14
A7
HIGHPIN
/BHE
U23_A14
(I/O)
U21_A14
INST
U20/U21_A14/A15
WDI
Ex_Lamp
Ex_Fan
BUZZER
CS510
Col 24
Col 23
Col 22
Col 21
Col 20
Col 19
Col 18
Col 17
Col 16
Col 15
Col 14
Col 13
Col EAB
Col 12
Col 11
Col 10
Col 9
Col 8
Col 7
Col 6
Col 5
Col 4
Col 3
Col 2
Col 1
Any Col
AoD
Any Row
Row A
Row B
Row C


AdD
Any Row
Any Col
Col 1
Col 2
Col 3
Col 4
Col 5
Col 6
Col 7
Col 8
Col 9
Col 10
Col 11
Col 12
Col EAB
Col 13
Col 14
Col 15
Col 16
Col 17
Col 18
Col 19
Col 20
Col 21
Col 22
Col 23
Col 24
Row A
Row B
Row C
Row D
Row E
Row F
INSTOUT
/WR
P2_1
A1
A2
A3
A4
A5
A6
A7
A8
A9
A10
A11
A12
A13
A14
A15
PLED5A1
PLED5A2
PLED6A1
PLED6A2
PLED7A1
PLED7A2
PLED8A1
PLED8A2
PLED8A2
DIPSW_9
DIPSW_9
DIPSW_10
DIPSW_11
DIPSW_12
DIPSW_13
DIPSW_14
DIPSW_15
count5_clk
count6_clk
count7_clk
count8_clk
PB_NOZZEL
RL5
RL6
RL7
RL8
SPARE5
SPARE6
SPARE7
SPARE8
LEDBANK
INT510
DB91_9_R
Data_In2
Data_In3
Data_In4
Data_In5
Data_In6
Data_In7
Data_In8
Data_In9
Data_In10
Data_In11
Data_In12
Data_In13
Data_In14
Data_In15
/RD
BUSMDTH
nWATN
nWRL
nWRH
A0
LJRXD
P2TXD
DB92_1_DCD
P2SXD
PB_NOZZEL
PS_NOZZEL
P7_NOZZEL
CLKOUT
P6_NOZZEL
ALE
P2_0
Data_In1
Data_In0
DB92_5_SGND
DB92_9_RI
DB91_5_SGND
CSRELAY
P1TXD
/WT
DB92_7_RTS
DB92_4_DTR
DB91_1_DCD
UTXD
DB91_8_CTS
/IRI
DB91_4_DTR
INST
DB92_6_DSR
RESET
DB91_7_RTS
DB92_8_CTS
DB91_6_DSR
P1RXD

Appendix Q

Routing of the FLEX 10Ks Internal Design


100 (DATA0)
107 (DCLK)
106 (nCE)
105 (TDI)
104 (GNDIO)
103 (GNDINT)
102 Data_in0
101 TEST_CPLD
100 U24_A14/A15
99 U24_A13
98 U20/U21_A13
97 U22_A14
96 U22/U23_A14/A15
95 U20_A14
94 (VCCIO)
93 (VCCINT)
92 A7
91 HIGHPIN
90 BHE
89 U23_A14
88 (WO)
87 U21_A14
86 INST
85 (GNDIO)
84 (GNDINT)
83 U20/U21_A14/A15
82 WDI
81 Ex_Lamp
80 Ex_Fan
79 BUZZER
78 CS510
77 (MSEL0)
76 (MSEL1)
75 (VCCINT)
74 (nCONFIG)
73 CE2
72 CE1
71 (VCCIO)
70 CE1
69 SPARE4
68 SPARE3
67 SPARE2
66 (GNDIO)
65 SPARE1
64 RL4
63 RL3
62 RL2
61 (VCCIO)
60 RL1
59 P4_NOZZEL
58 (GNDINT)
57 (GNDINT)
56 P3_NOZZEL
55 P2_NOZZEL
54 P1_NOZZEL
53 (VCCINT)
52 (VCCINT)
51 count_clk
50 (GNDIO)
49 count_clk
48 count2_clk
47 count1_clk
46 DIPSW_7
45 (VCCIO)
44 DIPSW_6
43 DIPSW_5
42 DIPSW_4
41 DIPSW_3
40 (GNDIO)
39 DIPSW_2
38 DIPSW_1
37 DIPSW_0
1 (TCK)
2 (CONF_DONE)
3 (nCEO)
4 (TDO)
5 (VCCIO)
6 (VCCINT)
7 A0
8 A1
9 A2
10 A3
11 A4
12 A5
13 A6
14 INIT_DONE
15 (GNDIO)
16 (GNDINT)
17 A8
18 A9
19 A10
20 A11
21 A12
22 A13
23 A14
24 (VCCIO)
25 (VCCINT)
26 A15
27 PLED1A1
28 PLED1A2
29 PLED2A1
30 PLED2A2
31 PLED3A1
32 PLED3A2
33 PLED4A1
34 (TMS)
35 (nSTATUS)
36 PLED4A2
109 Data_in1
110 Data_in2
111 Data_in3
112 Data_in4
113 Data_in5
114 Data_in6
115 (VCCIO)
116 Data_in7
117 Data_in8
118 Data_in9
119 Data_in10
120 Data_in11
121 Data_in12
122 Data_in13
123 (VCCINT)
124 ARESET
125 CLKOUT
126 ALE
127 (GNDINT)
128 Data_in14
129 (GNDIO)
130 Data_in15
131 RD
132 BUSWIDTH
133 MVATN
134 (VCCIO)
135 MRL
136 U22/U23_A13
137 MRH
138 LEDBANK
139 (GNDIO)
140 PORT2_RI
141 OUT0
142 U24_A14
143 HLDA
144 MR


1 (TCK)
2 (CONF_DONE)
3 (nCE0)
4 (TDO)
5 (VCCIO)
6 (VCCINT)
7 P2_1
8 A1
9 A2
10 A3
11 A4
12 A5
13 A6
14 A7
15 (GNDIO)
16 (GNDINT)
17 A8
18 A9
19 A10
20 A11
21 A12
22 A13
23 A14
24 (VCCIO)
25 (VCCINT)
26 A15
27 PLED5A1
28 PLED5A2
29 PLED6A1
30 PLED6A2
31 PLED7A1
32 PLED7A2
33 PLED8A1
34 (TMS)
35 (nSTATUS)
36 PLED8A2
37 DIPSW_8
38 DIPSW_9
39 DIPSW_10
40 (GNDIO)
41 DIPSW_11
42 DIPSW_12
43 DIPSW_13
44 DIPSW_14
45 (VCCIO)
46 DIPSW_15
47 count5_clk
48 count6_clk
49 count7_clk
50 (GNDIO)
51 count8_clk
52 (VCCINT)
53 (VCCINT)
54 P5_NOZZEL
55 P6_NOZZEL
56 P7_NOZZEL
57 (GNDINT)
58 (GNDINT)
59 P8_NOZZEL
60 RL5
61 (VCCIO)
62 RL6
63 RL7
64 RL8
65 SPARE5
66 (GNDIO)
67 SPARE6
68 SPARE7
69 SPARE8
70 LEDBANK
71 (VCCIO)
72 INT510
73 D891_9_RI
74 (nCONFIG)
75 (VCCINT)
76 (MSEL1)
77 (MSEL0)
78 P1RXD
79 D891_6_DSR
80 D892_8_CTS
81 D891_7_RTS
82 RESET
83 D892_6_DSR
84 (GNDINT)
85 (GNDIO)
86 INST
87 D891_4_DTR
88 LRI
89 D891_8_CTS
90 UTXD
91 D891_1_DCD
92 D892_4_DTR
93 (VCCINT)
94 (VCCIO)
95 D892_7_RTS
96 ANT
97 P1TXD
98 CSRELAY
99 D891_5_SGND
100 D892_9_RI
101 D892_5_SGND
102 Data_in0
103 (GNDINT)
104 (GNDIO)
105 (TDI)
106 (nCE)
107 (DCLK)
108 (DATA0)
109 Data_in1
110 Data_in2
111 Data_in3
112 Data_in4
113 Data_in5
114 Data_in6
115 (VCCIO)
116 Data_in7
117 Data_in8
118 Data_in9
119 Data_in10
120 Data_in11
121 Data_in12
122 Data_in13
123 (VCCINT)
124 P2_0
125 CLKOUT
126 ALE
127 (GNDINT)
128 Data_in14
129 (GNDIO)
130 Data_in15
131 nRD
132 BUSWIDTH
133 nWAITN
134 (VCCIO)
135 nWRL
136 A0
137 nWRH
138 P2RXD
139 (GNDIO)
140 D892_1_DCD
141 P2TXD
142 URXD
143 INSTOUT
144 WR

N.C. = No Connect, This pin has no internal connection to the device.
VCCINT = Dedicated power pin, which MUST be connected to VCC (5.0 volts).
VCCIO = Dedicated power pin, which MUST be connected to VCC (5.0 volts).
DINT = Dedicated ground pin or unused dedicated input, which MUST be connected to GND.
GNDIO = Dedicated ground pin, which MUST be connected to GND.
RESERVED = Unused I/O pin, which MUST be left unconnected.

CHIP "petscha1" ASSIGNED TO AN EPF10K10TC144-4

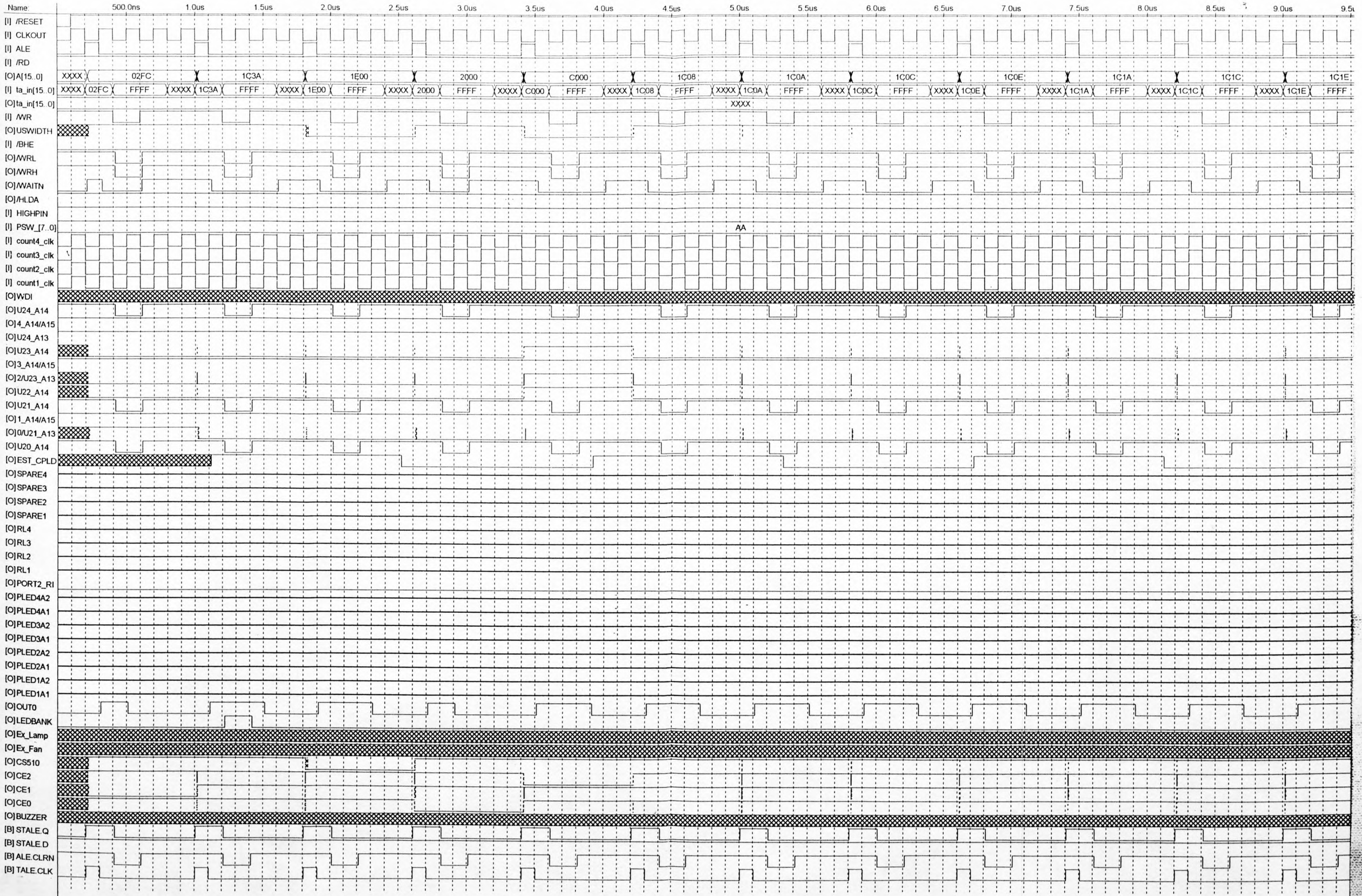
TCK	: 1
CONF_DONE	: 2
nCEO	: 3
TDO	: 4
VCCIO	: 5
VCCINT	: 6
A0	: 7
A1	: 8
A2	: 9
A3	: 10
A4	: 11
A5	: 12
A6	: 13
INIT_DONE	: 14
GNDIO	: 15
GNDINT	: 16
A8	: 17
A9	: 18
A10	: 19
A11	: 20
A12	: 21
A13	: 22
A14	: 23
VCCIO	: 24
VCCINT	: 25
A15	: 26
PLED1A1	: 27
PLED1A2	: 28
PLED2A1	: 29
PLED2A2	: 30
PLED3A1	: 31
PLED3A2	: 32
PLED4A1	: 33
TMS	: 34
nSTATUS	: 35
PLED4A2	: 36
DIPSW_0	: 37
DIPSW_1	: 38
DIPSW_2	: 39
GNDIO	: 40

DIPSW_3 : 41
DIPSW_4 : 42
DIPSW_5 : 43
DIPSW_6 : 44
VCCIO : 45
DIPSW_7 : 46
count1_clk : 47
count2_clk : 48
count3_clk : 49
GNDIO : 50
count4_clk : 51
VCCINT : 52
VCCINT : 53
P1_NOZZEL : 54
P2_NOZZEL : 55
P3_NOZZEL : 56
GNDINT : 57
GNDINT : 58
P4_NOZZEL : 59
RL1 : 60
VCCIO : 61
RL2 : 62
RL3 : 63
RL4 : 64
SPARE1 : 65
 DIO : 66
SPARE2 : 67
SPARE3 : 68
SPARE4 : 69
CE1 : 70
VCCIO : 71
CE0 : 72
CE2 : 73
nCONFIG : 74
VCCINT : 75
MSEL1 : 76
MSEL0 : 77
CS510 : 78
BUZZER : 79
Ex_Fan : 80
Ex_Lamp : 81
WDI : 82
U20/U21_A14/A15 : 83
GNDINT : 84
GNDIO : 85
INST : 86
U?1_A14 : 87
. SERVED : 88
U23_A14 : 89
/BHE : 90
USEREADY : 91
A7 : 92
VCCINT : 93
VCCIO : 94
U20_A14 : 95
U22/U23_A14/A15 : 96
U22_A14 : 97
U20/U21_A13 : 98
U24_A13 : 99
U24_A14/A15 : 100
TEST_CPLD : 101
Data_in0 : 102
GNDINT : 103
GNDIO : 104
TDI : 105
nCE : 106
DCLK : 107
DATA0 : 108
Data_in1 : 109
Data_in2 : 110
Data_in3 : 111
Data_in4 : 112

```
Data_in5                        : 113
Data_in6                        : 114
VCCIO                           : 115
Data_in7                        : 116
Data_in8                        : 117
Data_in9                        : 118
Data_in10                       : 119
Data_in11                       : 120
Data_in12                       : 121
Data_in13                       : 122
VCCINT                          : 123
/RESET                          : 124
CLKOUT                          : 125
ALE                             : 126
GNDINT                          : 127
Data_in14                       : 128
GNDIO                           : 129
Data_in15                       : 130
/RD                             : 131
BUSWIDTH                        : 132
/WAITN                          : 133
VCCIO                           : 134
/WRL                            : 135
U22/U23_A13                     : 136
/WRH                            : 137
  DBANK                         : 138
GNDIO                           : 139
PORT2_RI                        : 140
OUT0                            : 141
U24_A14                         : 142
/HLDA                           : 143
/WR                             : 144
```

Appendix R

Simulation of the FLEX 10Ks Internal Design


Name:
500.0ns
1.0us
1.5us
2.0us
2.5us
3.0us
3.5us
4.0us
4.5us
5.0us
5.5us
6.0us
6.5us
7.0us
7.5us
8.0us
8.5us
9.0us
[I] /RESET
[I] CLKOUT
[I] ALE
[I] /RD
[O]A[15..0]
[I] ta_in[15..0]
[O]ta_in[15..0]
[I] /WR
[O]USWIDTH
[I] /BHE
[O]WRL
[O]WRH
[O]WAITN
[O]HLDA
[I] HIGHPIN
[I] PSW_[7..0]
[I] count4_clk
[I] count3_clk
[I] count2_clk
[I] count1_clk
[O]WDI
[O]U24_A14
[O]4_A14/A15
[O]U24_A13
[O]U23_A14
[O]3_A14/A15
[O]2/U23_A13
[O]U22_A14
[O]U21_A14
[O]1_A14/A15
[O]0/U21_A13
[O]U20_A14
[O]EST_CPLD
[O]SPARE4
[O]SPARE3
[O]SPARE2
[O]SPARE1
[O]RL4
[O]RL3
[O]RL2
[O]RL1
[O]PORT2_RI
[O]PLED4A2
[O]PLED4A1
[O]PLED3A2
[O]PLED3A1
[O]PLED2A2
[O]PLED2A1
[O]PLED1A2
[O]PLED1A1
[O]OUT0
[O]LEDBANK
[O]Ex_Lamp
[O]Ex_Fan
[O]CS510
[O]CE2
[O]CE1
[O]CE0
[O]BUZZER
[B] STALE.Q
[B] STALE.D
[B] ALE.CLRN
[B] TALE.CLK
XXXX
02FC
FFFF
1C3A
1E00
2000
C000
1C08
1C0A
1C0C
1C0E
1C1A
1C1C
1C1E
AA

MAX+plus II 9.21 File: D:\SCFS\PETSCHA1.SCF

Name:

1.0us 2.0us 3.0us 4.0us 5.0us 6.0us 7.0us 8.0us 9.0us 10.0us 11.0us 12.0us

[I] RESET
[I] CLKOUT
[I] ALE
[I] ta_in[15..8]: XX - FF - - FF - - FF - - FF - - FF - - FF - - FF - - FF - - FF - - FF - - FF - - FF - - 00
[I] A[15..0]: XXXX 1C00 1C04 1C06 1C10 1C12 1C14 1C16 1C22 1C24 1C26 1C28 1C34 1C00 0000
[I] WR
[I] RD
[O]ESCNTR8
[O]ESCNTR7
[O]ESCNTR6
[O]ESCNTR5
[O]NOZZLES
[O]CS510
[O]CSRELAY
[O]CELED
[O]E_CNTRS
[O]BIT16LED
[O]ENCNTR8
[O]ENCNTR7
[O]ENCNTR6
[O]ENCNTR5
[I] count8_clk
[I] count7_clk
[I] count6_clk
[I] count5_clk
[O]SPARE8
[O]SPARE7
[O]SPARE6
[O]SPARE5
[O]RL8
[O]RL7
[O]RL6
[O]RL5
[I] P2RXD

Name:	200.0ns	400.0ns	600.0ns	800.0ns	1.0us	1.2us	1.4us	1.6us	1.8us	2.0us	2.2us	2.4us	2.6us	2.8us	3.0us	3.2us	3.4us	3.6us	3.8us	4.0us	4.2us	4.4us	4.6us
[I] RESCNTR4																							
[I] RESCNTR3																							
[I] RESCNTR2																							
[I] RESCNTR1																							
[I] ENCNTR4																							
[I] ENCNTR3																							
[I] ENCNTR2																							
[I] ENCNTR1																							
[I] count4_clk																							
[I] count3_clk																							
[I] count2_clk																							
[I] count1_clk																							
[I] CE_CNTRS																							
[I] SEL[1..0]													0										
[O]Data_in[15..0]	0000	0001	0002	0003	0004	0005	0000	0001	0002	0003	0004	0005	0006	0007	0008	0009	000A	000B	000C	000D	000E	000F	0010
[B] ata_out[15..0]	0000	0001	0002	0003	0004	0005	0000	0001	0002	0003	0004	0005	0006	0007	0008	0009	000A	000B	000C	000D	000E	000F	0010
[B] |data0_[15..0]	0000	0001	0002	0003	0004	0005	0000	0001	0002	0003	0004	0005	0006	0007	0008	0009	000A	000B	000C	000D	000E	000F	0010
[B] |data1_[15..0]	0000	0001	0002	0003	0004	0000	0001	0002	0003	0004	0005	0006	0007	0008	0009	000A	000B	000C	000D	000E	000F	0010	0011
[B] |data2_[15..0]	0000	0001	0002	0003	0000	0001	0002	0003	0004	0005	0006	0007	0008	0009	000A	000B	000C	000D	000E	000F	0010	0011	0012
[B] |data3_[15..0]	0000	0001	0002	0000	0001	0002	0003	0004	0005	0006	0007	0008	0009	000A	000B	000C	000D	000E	000F	0010	0011	0012	0013
[I] mux:8|sel[1..0]													0										
[B] nter:1|q[15..0]	0000	0001	0002	0003	0004	0005	0000	0001	0002	0003	0004	0005	0006	0007	0008	0009	000A	000B	000C	000D	000E	000F	0010
[B] er:1|dffs[15..0]	0000	0001	0002	0003	0004	0005	0000	0001	0002	0003	0004	0005	0006	0007	0008	0009	000A	000B	000C	000D	000E	000F	0010

Appendix S

Software Timing Analysis Results using MAX+PLUS II

Delay Matrix

Delay in ns

	A[2]	A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]	A[8]	A[9]	A[10]	A[11]	A[12]	A[13]	A[14]	A[15]	/RAS	RA[7:0]	CE0	CE1	CE2	CLKOUT	count1.clk	count2.clk	count3.clk	count4.clk	CS[5:0]	Data a[0]	Data a[1]	Data a[2]	Data a[3]	Data a[4]	Data a[5]	Data a[6]	Data a[7]	Data a[8]	Data a[9]	Data a[10]	Data a[11]	Data a[12]	Data a[13]	Data a[14]	Data a[15]	DPSR.0	DPSR.1	DPSR.2	DPSR.3	DPSR.4	DPSR.5
A[2]		15.5ns		16.0ns	15.5ns	14.9ns	14.6ns	14.5ns	13.6ns	14.0ns	13.5ns	14.0ns	14.0ns	14.0ns	14.0ns	14.6ns	14.2ns		15.3ns/31.3ns	22.9ns/43.5ns	17.8ns/38.8ns	18.9ns/25.5ns	17.3ns/19.1ns						19.4ns/32.4ns	27.3ns/54.3ns	27.6ns/54.3ns	27.6ns/55.1ns	27.6ns/55.6ns	27.6ns/56.3ns	27.6ns/53.3ns	27.6ns/57.3ns	27.5ns/57.3ns	26.2ns/55.4ns	26.2ns/53.7ns	26.2ns/53.8ns	26.2ns/53.7ns	26.2ns/53.5ns	26.2ns/53.5ns	26.2ns/54.8ns	25.1ns/55.2ns	25.2ns/54.5ns		?					
A0																																																			
A1																																																			
A2																																																			
A3																																																			
A4																																																			
A5																																																			
A6																																																			
A7																																																			
A8																																																			
A9																																																			
A10																																																			
A11																																																			
A12																																																			
A13																																																			
A14																																																			
A15																																																			
/OE																																																			
RA[7:0]																																																			
RA[7:0]																																																			
CE0																																																			
CE1																																																			
CE2																																																			
CLKOUT		10.0ns		16.7ns	16.0ns	15.6ns	15.1ns	15.2ns	13.4ns	14.0ns	13.9ns	14.0ns	14.0ns	14.0ns	14.0ns	14.6ns	14.2ns		15.3ns/31.3ns	23.6ns/43.5ns	17.8ns/38.8ns	18.9ns/25.5ns	17.3ns/19.1ns						19.4ns/32.4ns	28.0ns/54.3ns	28.1ns/54.8ns	28.1ns/55.1ns	28.1ns/55.6ns	28.1ns/56.3ns	28.1ns/53.3ns	28.1ns/57.3ns	28.2ns/57.3ns	26.9ns/55.4ns	26.9ns/53.7ns	26.9ns/53.8ns	26.9ns/53.7ns	26.9ns/53.5ns	26.9ns/53.5ns	26.9ns/54.8ns	25.8ns/55.2ns	25.7ns/54.5ns							
count1.clk																														27.8ns	28.2ns	28.1ns	28.4ns	28.1ns	27.4ns	27.4ns	27.4ns	26.2ns	26.1ns	25.1ns	25.1ns	25.1ns	24.1ns	24.1ns	24.1ns	24.1ns							
count2.clk																														26.0ns	24.6ns	24.6ns	24.6ns	24.8ns	24.6ns	24.6ns	24.6ns	24.5ns	24.3ns	24.1ns	24.1ns	24.1ns	24.1ns	24.1ns	24.1ns	24.1ns							
count3.clk																														25.3ns	23.0ns	23.0ns	24.0ns	23.0ns	23.0ns	23.0ns	27.8ns	25.1ns	26.6ns	24.1ns	24.1ns	24.1ns	24.1ns	24.1ns	24.1ns	24.1ns							
count4.clk																														27.1ns	23.7ns	23.7ns	23.6ns	23.6ns	23.6ns	23.6ns	27.8ns	25.1ns	26.6ns	22.7ns	22.7ns	22.5ns	24.4ns	24.4ns	24.0ns	24.4ns							
CS[5:0]																																																			
Data a[0]		17.3ns					1													20.8ns																																
Data a[1]																				25.6ns										30.0ns/49.7ns	30.3ns/47.8ns	30.3ns/47.8ns	30.3ns/48.0ns	30.3ns/48.0ns	30.3ns/47.7ns	30.3ns/49.4ns	30.2ns/49.5ns	28.8ns/48.0ns	28.8ns/47.1ns	28.8ns/47.2ns	28.8ns/47.1ns	28.9ns/47.2ns	28.8ns/48.2ns	27.8ns/48.6ns	27.7ns/48.3ns							
Data a[2]			18.2ns																	25.1ns										29.5ns/49.2ns	29.8ns/47.3ns	29.8ns/47.3ns	29.8ns/47.5ns	29.8ns/47.5ns	29.8ns/47.7ns	29.8ns/48.5ns	29.7ns/49.0ns	28.4ns/48.3ns	28.4ns/46.6ns	28.4ns/48.2ns	28.4ns/46.6ns	28.4ns/46.7ns	28.4ns/47.3ns	27.3ns/48.1ns	27.2ns/47.8ns							
Data a[3]				17.4ns																27.7ns										29.1ns/48.5ns	30.0ns/39.0ns	30.0ns/38.8ns	30.0ns/39.2ns	30.0ns/38.3ns	30.0ns/41.1ns	30.0ns/41.1ns	29.9ns/41.1ns	28.8ns/39.6ns	28.6ns/37.5ns	28.8ns/38.0ns	28.6ns/37.5ns	28.6ns/37.2ns	28.6ns/39.0ns	27.5ns/39.1ns	27.4ns/39.1ns							
Data a[4]					17.3ns															30.0ns										29.3ns/43.3ns	30.0ns/41.4ns	30.0ns/41.4ns	30.0ns/41.6ns	30.0ns/41.6ns	30.0ns/41.6ns	30.0ns/43.6ns	29.5ns/43.1ns	28.6ns/42.4ns	28.6ns/40.7ns	28.6ns/40.8ns	28.6ns/40.7ns	28.6ns/40.6ns	28.6ns/40.8ns	27.5ns/42.2ns	27.4ns/41.5ns							
Data a[5]							17.5ns													30.0ns										32.0ns/43.5ns	32.3ns/42.0ns	32.3ns/42.2ns	32.3ns/42.2ns	32.3ns/41.9ns	32.3ns/43.6ns	32.2ns/43.7ns	32.2ns/43.7ns	30.9ns/43.0ns	30.8ns/41.3ns	30.9ns/41.4ns	30.8ns/41.3ns	30.9ns/41.2ns	30.8ns/42.4ns	29.8ns/42.0ns	29.7ns/42.5ns							
Data a[6]								18.6ns												32.6ns										34.3ns/45.2ns	35.8ns/43.5ns	35.8ns/43.5ns	35.8ns/44.1ns	35.8ns/44.6ns	35.8ns/46.0ns	35.8ns/46.6ns	35.8ns/46.6ns	33.6ns/44.6ns	33.6ns/42.4ns	33.6ns/42.5ns	33.6ns/42.3ns	33.6ns/43.5ns	32.5ns/43.5ns	32.4ns/43.6ns							
Data a[7]									16.6ns																					35.1ns/45.6ns	35.4ns/43.3ns	35.4ns/44.2ns	35.4ns/44.5ns	35.6ns/44.6ns	35.6ns/44.6ns	35.4ns/46.4ns	35.3ns/46.6ns	34.0ns/44.9ns	34.0ns/42.8ns	34.0ns/42.9ns	34.0ns/42.9ns	34.0ns/43.5ns	34.0ns/43.5ns	32.8ns/44.3ns	32.8ns/44.6ns							
Data a[8]										16.5ns										24.6ns										25.1ns	30.5ns/49.2ns	30.2ns/47.3ns	30.2ns/48.0ns	30.2ns/48.3ns	30.2ns/48.1ns	30.2ns/50.2ns	30.1ns/50.2ns	30.2ns/50.2ns	30.1ns/50.2ns	30.2ns/50.2ns	30.2ns/50.2ns	30.2ns/50.2ns	30.2ns/50.2ns	30.2ns/50.2ns	30.2ns/50.2ns	30.2ns/50.2ns							
Data a[9]											16.6ns									21.3ns	33.0ns		22.1ns							22.0ns	35.5ns/48.2ns	30.2ns/44.7ns	30.2ns/45.0ns	30.2ns/45.3ns	30.2ns/45.2ns	30.2ns/47.2ns	30.2ns/47.2ns	30.1ns/47.2ns	30.2ns/47.2ns	37.8ns/48.3ns	37.8ns/48.6ns	37.8ns/48.7ns	37.8ns/48.6ns	37.8ns/48.6ns	37.8ns/48.6ns	36.7ns/48.1ns	36.6ns/47.8ns							
Data a[10]												16.5ns								24.6ns	30.0ns		24.7ns							25.1ns	30.5ns/49.2ns	30.2ns/47.7ns	30.2ns/48.0ns	30.2ns/48.3ns	30.2ns/48.2ns	30.2ns/50.2ns	30.2ns/50.2ns	30.1ns/50.2ns	30.2ns/50.2ns	37.8ns/48.3ns	37.8ns/48.6ns	37.8ns/48.7ns	37.8ns/48.6ns	37.8ns/48.6ns	37.8ns/48.6ns	36.7ns/48.1ns	36.6ns/47.8ns							
Data a[11]													16.5ns							24.1ns	30.3ns		24.7ns							24.8ns	30.4ns/48.7ns	30.2ns/47.3ns	30.2ns/47.5ns	30.2ns/47.8ns	30.2ns/47.8ns	30.2ns/49.7ns	30.2ns/49.7ns	30.2ns/49.8ns	30.2ns/49.7ns	37.3ns/47.8ns	37.3ns/48.1ns	37.3ns/48.2ns	37.3ns/48.1ns	37.3ns/48.0ns	37.3ns/48.2ns	36.2ns/47.8ns	36.1ns/47.5ns							
Data a[12]														18.4ns						26.9ns	39.1ns		21.3ns/24.1ns							27.6ns	41.2ns/51.5ns	41.5ns/50.0ns	41.5ns/50.3ns	41.5ns/50.6ns	41.5ns/50.5ns	41.5ns/52.5ns	41.5ns/52.5ns	41.5ns/52.5ns	41.4ns/52.5ns	40.1ns/50.6ns	40.1ns/49.9ns	40.1ns/49.9ns	40.1ns/49.8ns	40.1ns/50.0ns	39.0ns/50.4ns	38.9ns/50.1ns							
Data a[13]															17.4ns					34.2ns	40.4ns	21.0ns	26.9ns/31.1ns							34.5ns	48.5ns/58.0ns	40.8ns/57.3ns	48.8ns/57.6ns	48.8ns/57.3ns	48.8ns/57.8ns	48.8ns/58.0ns	48.8ns/58.0ns	48.8ns/58.0ns	47.4ns/57.9ns	47.4ns/56.2ns	47.4ns/56.3ns	47.4ns/56.2ns	47.4ns/54.1ns	47.4ns/54.1ns	47.4ns/57.3ns	46.3ns/57.3ns	46.2ns/57.4ns							
Data a[14]																17.5ns				21.3ns/34.4ns	40.6ns	20.7ns	27.1ns/31.6ns	22.0ns						35.1ns	48.7ns/58.0ns	49.0ns/57.5ns	49.0ns/57.8ns	49.0ns/58.0ns	49.0ns/58.1ns	49.0ns/58.0ns	49.0ns/60.0ns	49.0ns/60.0ns	48.9ns/60.0ns	47.6ns/58.1ns	47.6ns/56.5ns	47.6ns/56.6ns	47.6ns/54.1ns	47.6ns/54.1ns	47.6ns/57.5ns	46.5ns/57.5ns	46.4ns/57.6ns							
Data a[15]																	17.1ns			18.2ns/34.6ns	40.8ns	21.1ns	27.3ns/31.8ns	20.2ns						35.1ns	48.9ns/58.2ns	49.2ns/57.7ns	49.2ns/58.0ns	49.2ns/58.3ns	49.2ns/58.2ns	49.2ns/58.2ns	49.2ns/60.2ns	49.2ns/60.2ns	49.1ns/60.2ns	47.8ns/58.3ns	47.8ns/56.6ns	47.8ns/56.7ns	47.8ns/54.1ns	47.8ns/54.1ns	47.8ns/57.7ns	46.7ns/57.8ns	46.6ns/57.8ns							
DPSR.0																															21.5ns																				
DPSR.1																																24.6ns																			
DPSR.2																																	24.6ns																		
DPSR.3																																		24.5ns																	
DPSR.4																																			24.4ns																
DPSR.5																																				24.5ns															
DPSR.6																																					24.5ns														
DPSR.7																																						24.4ns													
Tx7ns																																																			
Tx Lamp																																																			
/Ack																																																			
MR_DONE																																																			
LED0old																																																			
OUT0																																																			
FLED1A1																																																			
FLED1A2																																																			
FLED1A3																																																			
FLED2A2																																																			
FLED2A3																																																			
FLED2A4																																																			
FLED3A2																																																			
FLED3A3																																																			
FLED3A4																																																			
FLED4A2																																																			
PORT.IR																																																			
P1.MUX[1]																														19.1ns																						
P2.MUX[1]																															21.5ns																				
P3.MUX[1]																																21.5ns																		
P4.MUX[1]																																	24.3ns																	
/RD																														25.1ns/35.4ns	25.4ns/33.5ns	25.4ns/34.1ns	25.4ns/34.4ns	25.4ns/34.3ns	25.4ns/36.1ns	25.4ns/36.5ns	25.3ns/36.3ns	24.0ns/34.5ns	24.0ns/32.8ns	24.0ns/32.5ns	24.0ns/32.8ns	24.0ns/32.7ns	24.0ns/33.9ns	22.9ns/34.3ns	22.8ns/34.0ns							
/RESET																																																			
RL1																																																				
RL2																																																				
RL3																																																			
RL4																																																			
SPARE1																																																			
SPARE2																																																			
SPARE3																																																			
SPARE4																																																			
TEST_CPLD																																																			
V20.A14																																																			
V20/A01.A13																																																			
V20/A01.A14/A15																																																			
V21.A14																																																			
V21.A14																																																			
V21.A13																																																			
V21.A14/A15																																																			
V21.A14																																																			
/WRH																																																			
WR																																																			
/WRH																																																			
/WRL																																																			

Delay Matrix

Destination

Source	DPSM.0	DPSM.7	L.x.Fan	L.x.Lamp	/MAIN	MB_DONE	LEDPWR	OUT0	PLED1A1	PLED1A2	PLED1A3	PLED1A4	PLED1A5	PLED1A6	PLED1A7	PLED1A8	PLED1A9	PLED1A10	PSM7.28	P1.MOVZEL	P2.MOVZEL	P3.MOVZEL	P4.MOVZEL	/MD	/AE SET	HL1	HL2	HL3	HL4	SPARE1	SPARE2	SPARE3	SPARE4	TEST_CPLD	M09.A14	M09/A01.A13	M09/A01.A14/A15	M21.A14	M27.A14	M27/A03.A13	M27/A03.A14/A15	M03.A14	M24.A13	M24.A14/A15	M21.A14	/MAIN	MB1	/MB	/MB1	/MB
A1			22.9ns/43.3ns	25.0ns/48.6ns			19.5ns/43.2ns		23.3ns/44.2ns	23.4ns/44.2ns	23.6ns/44.3ns	23.6ns/44.3ns	23.6ns/44.3ns	23.6ns/44.1ns	23.6ns/44.1ns	23.6ns/44.1ns	23.6ns/44.2ns	19.5ns/43.1ns							19.6ns/47.1ns	19.6ns/47.1ns	19.6ns/47.1ns	19.8ns/47.1ns	19.8ns/47.1ns	19.8ns/47.1ns	19.9ns/47.1ns	19.9ns/47.0ns		18.5ns	19.4ns/29.4ns		18.5ns	13.4ns	13.2ns	13.4ns					23.4ns/47.0ns		18.1ns	18.0ns
A0																																																	
A1																																																	
A2																																																
A3																																															
A4																																															
A5																																															
A6																																															
A7																																															
A8																																															
A9																																															
A10																																															
A11																																															
A12																																															
A13																																															
A14																																															
A15																																															
/RAE																																															
DCSmotor																																															
FAUTER																																															
C0.0																																															
C1.1																																															
C2.2																																															
CLKOUT			23.6ns/43.3ns	25.7ns/48.6ns			20.2ns/43.2ns	12.1ns	24.0ns/44.2ns	24.1ns/44.2ns	24.1ns/44.3ns	24.1ns/44.1ns	24.1ns/44.1ns	24.1ns/44.1ns	24.1ns/44.1ns	24.1ns/44.1ns	23.8ns/41.8ns	20.2ns/43.1ns							20.1ns/47.1ns	20.1ns/47.1ns	20.1ns/47.1ns	20.2ns/47.1ns	20.2ns/47.1ns	20.4ns/47.0ns	20.4ns/47.0ns	20.4ns/47.0ns	18.2ns/19.4ns	19.6ns	19.4ns/29.4ns			13.8ns	13.2ns	13.4ns				12.4ns	24.1ns/47.0ns		18.7ns
crout1.clk																																															
crout2.clk																																															
crout3.clk																																															
crout4.clk																																															
C22.10																																															
Data a0			20.6ns	20.6ns			1		20.3ns																15.7ns									20.1ns										20.6ns		20.0ns
Data a1			25.6ns	27.8ns			22.2ns		30.1ns/40.6ns	30.2ns/40.6ns	30.2ns/40.5ns	30.2ns/40.5ns	30.2ns/40.5ns	30.2ns/40.7ns	30.2ns/40.7ns	30.2ns/40.7ns	37.9ns/38.2ns	22.3ns							32.1ns/38.5ns	15.8ns/38.5ns	32.1ns/38.5ns	32.3ns/38.5ns	32.3ns/38.5ns	32.3ns/38.6ns	32.4ns/38.6ns	32.4ns/38.6ns														20.6ns		
Data a2			25.1ns	27.2ns			21.7ns		27.8ns/40.1ns	32.3ns/40.1ns	37.3ns/40.2ns	37.3ns/40.0ns	16.2ns/40.2ns	37.3ns/40.0ns	37.3ns/40.2ns	37.3ns/40.2ns	32.4ns/37.7ns	21.3ns							31.5ns/35.9ns	31.5ns/35.9ns	15.8ns/35.9ns	31.7ns/35.9ns	31.7ns/35.9ns	31.8ns/35.8ns	31.8ns/35.8ns	31.8ns/35.8ns														25.6ns		
Data a3			27.7ns	32.7ns			27.0ns		25.4ns/27.1ns	25.5ns/27.1ns	25.5ns/27.0ns	25.5ns/27.0ns	25.5ns/27.0ns	25.5ns/27.1ns	10.2ns/20.0ns	25.2ns/25.5ns	26.9ns/26.9ns	26.9ns							26.4ns/30.0ns	26.4ns/30.0ns	26.4ns/30.0ns	15.7ns/30.0ns	26.8ns/30.0ns	26.7ns/30.0ns	26.7ns/30.0ns	26.7ns/30.0ns														30.0ns		
Data a4			30.5ns	28.2ns			30.0ns		31.3ns/34.2ns	18.2ns/34.2ns	31.8ns/34.1ns	31.8ns/34.1ns	31.8ns/34.1ns	31.8ns/34.1ns	31.8ns/34.1ns	31.8ns/34.1ns	31.5ns/33.1ns	30.1ns							32.1ns/38.5ns	32.1ns/38.5ns	32.1ns/38.5ns	32.3ns/38.5ns	15.7ns/38.5ns	32.4ns/38.6ns	32.4ns/38.6ns	32.4ns/38.6ns														34.0ns		
Data a5			30.0ns	33.1ns			29.1ns		32.3ns/34.0ns	32.4ns/34.0ns	32.4ns/34.0ns	16.1ns/34.0ns	32.4ns/34.0ns	32.4ns/34.0ns	32.4ns/34.0ns	32.4ns/34.0ns	32.1ns/32.4ns	29.0ns							27.5ns/29.5ns	27.5ns/29.5ns	27.5ns/29.5ns	27.7ns/29.5ns	15.0ns/29.5ns	27.8ns/29.6ns	27.8ns/29.6ns														33.1ns		
Data a6			32.6ns	37.3ns			31.1ns		30.6ns/32.9ns	30.5ns/32.9ns	30.5ns/33.0ns	30.5ns/33.0ns	30.5ns/33.0ns	18.2ns/32.8ns	30.5ns/33.0ns	30.5ns/33.0ns	30.3ns/30.5ns	31.8ns							31.4ns/35.8ns	31.4ns/35.8ns	31.4ns/35.8ns	31.8ns/35.8ns	31.8ns/35.8ns	31.7ns/35.8ns	31.7ns/35.8ns														35.7ns		
Data a7			33.0ns	37.7ns			32.1ns		30.8ns/33.3ns	30.8ns/33.3ns	30.9ns/33.4ns	30.9ns/33.2ns	30.9ns/33.4ns	30.9ns/33.4ns	30.9ns/33.4ns	30.9ns/33.4ns	15.5ns/30.9ns	32.2ns							31.8ns/38.2ns	31.8ns/38.2ns	31.8ns/38.2ns	32.0ns/38.2ns	32.0ns/38.2ns	32.1ns/38.1ns	32.1ns/38.1ns	15.5ns/38.1ns														36.1ns		
Data a8			36.0ns	41.5ns			36.1ns		34.6ns/37.1ns	34.7ns/37.1ns	34.7ns/37.2ns	34.7ns/37.2ns	34.7ns/37.2ns	34.7ns/37.2ns	34.7ns/37.2ns	34.7ns/37.2ns	34.4ns/34.7ns	36.0ns							35.6ns/40.0ns	35.6ns/40.0ns	35.6ns/40.0ns	35.8ns/40.0ns	35.8ns/40.0ns	35.8ns/40.0ns	35.8ns/40.0ns	35.9ns/39.9ns														39.9ns		
Data a9			33.0ns	38.5ns			33.1ns		31.8ns/34.1ns	31.9ns/34.1ns	31.9ns/34.2ns	31.9ns/34.2ns	31.9ns/34.2ns	31.9ns/34.2ns	31.9ns/34.2ns	31.9ns/34.2ns	31.4ns/31.9ns	33.0ns							32.6ns/37.0ns	32.6ns/37.0ns	32.6ns/37.0ns	32.8ns/37.0ns	32.8ns/37.0ns	32.8ns/37.0ns	32.9ns/36.9ns	32.9ns/36.9ns														36.9ns		
Data a10			36.0ns	41.5ns			36.1ns		34.6ns/37.1ns	34.7ns/37.1ns	34.7ns/37.2ns	34.7ns/37.2ns	34.7ns/37.2ns	34.7ns/37.2ns	34.7ns/37.2ns	34.7ns/37.2ns	34.4ns/34.7ns	36.0ns							35.6ns/40.0ns	35.6ns/40.0ns	35.6ns/40.0ns	35.8ns/40.0ns	35.8ns/40.0ns	35.8ns/39.9ns	35.9ns/39.9ns	35.9ns/39.9ns														39.9ns		
Data a11			36.3ns	41.0ns			35.6ns		34.1ns/36.6ns	34.2ns/36.8ns	34.2ns/36.7ns	34.2ns/36.5ns	34.2ns/36.5ns	34.2ns/36.5ns	34.2ns/36.5ns	34.2ns/36.5ns	33.9ns/34.2ns	35.5ns							35.1ns/39.5ns	35.1ns/39.5ns	35.1ns/39.5ns	35.3ns/39.5ns	35.3ns/39.5ns	35.4ns/39.6ns	35.4ns/39.6ns	35.4ns/39.6ns														39.6ns		
Data a12			39.1ns	43.8ns			38.4ns		36.0ns/39.1ns	37.0ns/39.4ns	37.0ns/39.5ns	37.0ns/39.3ns	37.0ns/39.5ns	37.0ns/39.5ns	37.0ns/39.5ns	37.0ns/39.5ns	36.7ns/37.0ns	38.3ns							37.0ns/42.3ns	37.5ns/42.3ns	37.5ns/42.3ns	38.1ns/42.3ns	38.1ns/42.3ns	38.2ns/42.2ns	38.2ns/42.2ns	38.2ns/42.2ns														42.2ns		
Data a13			46.4ns	51.1ns			45.7ns		44.2ns/46.3ns	44.3ns/46.7ns	44.3ns/46.6ns	44.3ns/46.6ns	44.3ns/46.6ns	44.3ns/46.6ns	44.3ns/46.6ns	44.3ns/46.6ns	44.0ns/44.3ns	45.6ns							45.2ns/49.6ns	45.2ns/49.6ns	45.2ns/49.6ns	45.4ns/49.6ns	45.4ns/49.6ns	45.5ns/49.6ns	45.5ns/49.6ns	45.5ns/49.5ns														49.5ns		
Data a14			46.8ns	51.3ns			45.8ns		44.4ns/46.8ns	44.5ns/46.8ns	44.5ns/46.8ns	44.5ns/47.0ns	44.5ns/46.8ns	44.5ns/47.0ns	44.5ns/46.8ns	44.5ns/47.0ns	44.0ns/44.5ns	45.8ns							45.4ns/49.8ns	45.4ns/49.8ns	45.4ns/49.8ns	45.6ns/49.8ns	45.6ns/49.8ns	45.6ns/49.8ns	45.6ns/49.8ns	45.6ns/49.6ns														49.6ns		
Data a15			46.8ns	51.5ns			46.1ns		44.8ns/47.1ns	44.7ns/47.1ns	44.7ns/47.2ns	44.7ns/47.2ns	44.7ns/47.2ns	44.7ns/47.2ns	44.7ns/47.2ns	44.7ns/47.2ns	44.4ns/44.7ns	46.0ns							45.6ns/50.0ns	45.6ns/50.0ns	45.6ns/50.0ns	45.8ns/50.0ns	45.8ns/50.0ns	45.8ns/49.9ns	45.9ns/49.9ns	45.9ns/49.9ns														49.9ns		
DPSM.0																																															
DPSM.1																																															
DPSM.2																																															
DPSM.3																																															
DPSM.4																																															
DPSM.5																																															
DPSM.6																																															
DPSM.7																																															
L.x.Fan																																															
L.x.Lamp																																															
/MAIN																																															
MB_DONE																																															
LEDPWR																																															
OUT0																																															
PLED1A1																																															
PLED1A2																																															
PLED1A3																																															
PLED1A4																																															
PLED1A5																																															
PLED1A6																																															
PLED1A7																																															
PLED1A8																																															
PLED1A9																																															
PLED1A10																																															
PSM7.28																																															
P1.MOVZEL																																															
P2.MOVZEL																																															
P3.MOVZEL																																															
P4.MOVZEL																																															
/MD																																															
/AE SET																																															
HL1																																															
HL2																																															
HL3																																															
HL4																																															
SPARE1																																															
SPARE2																																															
SPARE3																																															
SPARE4																																															
TEST_CPLD																																															
M09.A14																																															
M09/A01.A13																																															
M09/A01.A14/A15																																														
M21.A14																																															
M27.A14																																															
M27/A03.A13																																															
M27/A03.A14/A15																																														
M03.A14																																															
M24.A13																																															
M24.A14/A15																																															
M21.A14																																															
/MAIN																																															
MB1																																															
/MB			22.6ns	20.1ns			21.3ns		21.0ns/23.5ns	21.1ns/23.5ns	21.1ns/23.6ns	21.1ns/23.6ns	21.1ns/23.6ns	21.1ns/23.6ns	21.1ns/23.6ns	21.1ns/23.6ns	20.8ns/21.1ns	21.8ns							20.8ns/20.3ns	20.8ns/20.3ns	20.8ns/20.3ns	20.8ns/20.3ns	20.8ns/20.3ns	20.7ns/20.2ns	20.7ns/20.2ns	20.7ns/20.2ns		17.2ns			18.1ns						15.2ns		25.3ns		15.1ns	15.0ns
/MB1																																															
/MB																																															

Delay Matrix

Destination

Source	CSRELAY	Data_in0	Data_in1	Data_in2	Data_in3	Data_in4	Data_in5	Data_in6	Data_in7	Data_in8	Data_in9	Data_in10	Data_in11	Data_in12	Data_in13	Data_in14	Data_in15	DB91_5_SGND	DB91_7_RTS	DB92_5_SGND	DB92_7_RTS	DB92_9_RI	/INIT	INSTOUT	LEDBANK
ALE																									
A0																									
A1	21.9ns	25.7ns/44.6ns	25.8ns/44.7ns	26.9ns/43.9ns	26.3ns/44.1ns	26.3ns/44.0ns	26.3ns/44.0ns	26.3ns/44.5ns	26.3ns/43.6ns	26.2ns/44.1ns	26.8ns/43.9ns	26.7ns/43.8ns	26.6ns/43.9ns	26.5ns/44.2ns	25.6ns/43.2ns	25.6ns/43.4ns	25.4ns/43.4ns								28.8ns
A2	24.2ns	22.1ns/41.2ns	22.2ns/41.3ns	23.3ns/40.5ns	22.7ns/40.7ns	22.7ns/40.6ns	22.7ns/40.6ns	22.7ns/41.1ns	22.7ns/40.2ns	22.6ns/40.7ns	23.2ns/40.5ns	23.1ns/40.4ns	23.0ns/40.5ns	22.9ns/40.8ns	22.0ns/39.8ns	22.0ns/40.0ns	21.8ns/40.0ns								30.1ns
A3	24.0ns	26.2ns/41.5ns	26.3ns/41.6ns	27.4ns/40.7ns	26.8ns/41.7ns	26.8ns/41.6ns	26.8ns/41.6ns	26.8ns/42.1ns	26.8ns/41.2ns	26.7ns/41.7ns	27.3ns/41.5ns	27.2ns/41.5ns	27.1ns/41.6ns	27.0ns/41.8ns	26.1ns/40.6ns	26.1ns/40.2ns	25.9ns/40.2ns								30.1ns
A4	17.3ns	29.4ns/44.7ns	29.5ns/44.8ns	30.6ns/43.9ns	30.0ns/44.9ns	30.0ns/44.8ns	30.0ns/44.8ns	30.0ns/45.3ns	30.0ns/44.4ns	26.2ns/44.9ns	26.2ns/44.7ns	26.2ns/44.7ns	26.1ns/44.8ns	26.0ns/45.0ns	25.1ns/43.8ns	25.1ns/43.4ns	24.7ns/43.4ns								24.2ns
A5	16.8ns	31.4ns/46.7ns	31.5ns/46.8ns	32.6ns/45.9ns	32.0ns/46.9ns	32.0ns/46.8ns	32.0ns/46.8ns	32.0ns/47.3ns	32.0ns/46.4ns	26.3ns/46.9ns	26.2ns/46.7ns	26.2ns/46.7ns	26.1ns/46.8ns	26.0ns/47.0ns	25.1ns/45.8ns	25.1ns/45.4ns	24.7ns/45.4ns								24.2ns
A6	28.7ns	33.3ns/48.6ns	33.4ns/48.7ns	34.5ns/47.8ns	33.9ns/48.8ns	33.9ns/48.7ns	33.9ns/48.7ns	33.9ns/49.2ns	33.9ns/48.3ns	33.8ns/48.8ns	34.4ns/48.6ns	34.3ns/48.6ns	34.2ns/48.7ns	34.1ns/48.9ns	33.2ns/47.7ns	33.2ns/47.3ns	33.0ns/47.3ns								35.1ns
A7	28.7ns	33.3ns/48.6ns	33.4ns/48.7ns	34.5ns/47.8ns	33.9ns/48.8ns	33.9ns/48.7ns	33.9ns/48.7ns	33.9ns/49.2ns	33.9ns/48.3ns	33.8ns/48.8ns	34.4ns/48.6ns	34.3ns/48.6ns	34.2ns/48.7ns	34.1ns/48.9ns	33.2ns/47.7ns	33.2ns/47.3ns	33.0ns/47.3ns								35.1ns
A8	34.1ns	23.0ns/54.0ns	23.1ns/54.1ns	23.1ns/53.2ns	23.1ns/54.2ns	23.1ns/54.1ns	23.1ns/54.1ns	23.1ns/54.6ns	23.1ns/53.7ns	39.2ns/54.2ns	39.8ns/54.0ns	39.7ns/54.0ns	39.6ns/54.1ns	39.5ns/54.3ns	38.6ns/53.1ns	38.6ns/52.7ns	38.4ns/52.7ns								40.5ns
A9	32.5ns	20.7ns/52.4ns	20.8ns/52.5ns	20.8ns/51.6ns	20.8ns/52.6ns	20.8ns/52.5ns	20.8ns/52.5ns	20.8ns/53.0ns	20.8ns/52.1ns	37.6ns/52.6ns	38.2ns/52.4ns	38.1ns/52.4ns	38.0ns/52.5ns	37.9ns/52.7ns	37.0ns/51.5ns	37.0ns/51.1ns	36.8ns/51.1ns								38.9ns
A10	34.1ns	23.0ns/54.0ns	23.1ns/54.1ns	23.1ns/53.2ns	23.1ns/54.2ns	23.1ns/54.1ns	23.1ns/54.1ns	23.1ns/54.6ns	23.1ns/53.7ns	39.2ns/54.2ns	39.8ns/54.0ns	39.7ns/54.0ns	39.6ns/54.1ns	39.5ns/54.3ns	38.6ns/53.1ns	38.6ns/52.7ns	38.4ns/52.7ns								40.5ns
A11	34.1ns	23.0ns/54.0ns	23.1ns/54.1ns	23.1ns/53.2ns	23.1ns/54.2ns	23.1ns/54.1ns	23.1ns/54.1ns	23.1ns/54.6ns	23.1ns/53.7ns	39.2ns/54.2ns	39.8ns/54.0ns	39.7ns/54.0ns	39.6ns/54.1ns	39.5ns/54.3ns	38.6ns/53.1ns	38.6ns/52.7ns	38.4ns/52.7ns								40.5ns
A12	36.5ns	25.4ns/56.4ns	25.5ns/56.5ns	25.5ns/55.6ns	25.5ns/56.6ns	25.5ns/56.5ns	25.5ns/56.5ns	25.5ns/57.0ns	25.5ns/56.1ns	41.6ns/56.6ns	42.2ns/56.4ns	42.1ns/56.4ns	42.0ns/56.5ns	41.9ns/56.7ns	41.0ns/55.5ns	41.0ns/55.1ns	40.8ns/55.1ns								42.9ns
A13	36.0ns	24.9ns/55.9ns	25.0ns/56.0ns	25.0ns/55.1ns	25.0ns/56.1ns	25.0ns/56.0ns	25.0ns/56.0ns	25.0ns/56.5ns	25.0ns/55.6ns	41.1ns/56.1ns	41.7ns/55.9ns	41.6ns/55.9ns	41.5ns/56.0ns	41.4ns/56.2ns	40.5ns/55.0ns	40.5ns/54.6ns	40.3ns/54.6ns								42.4ns
A14	36.5ns	25.4ns/56.4ns	25.5ns/56.5ns	25.5ns/55.6ns	25.5ns/56.6ns	25.5ns/56.5ns	25.5ns/56.5ns	25.5ns/57.0ns	25.5ns/56.1ns	41.6ns/56.6ns	42.2ns/56.4ns	42.1ns/56.4ns	42.0ns/56.5ns	41.9ns/56.7ns	41.0ns/55.5ns	41.0ns/55.1ns	40.8ns/55.1ns								42.9ns
A15	40.1ns	29.0ns/60.0ns	29.1ns/60.1ns	29.1ns/59.2ns	29.1ns/60.2ns	29.1ns/60.1ns	29.1ns/60.1ns	29.1ns/60.6ns	29.1ns/59.7ns	45.2ns/60.2ns	45.8ns/60.0ns	45.7ns/60.0ns	45.6ns/60.1ns	45.5ns/60.3ns	44.6ns/59.1ns	44.6ns/58.7ns	44.4ns/58.7ns								46.5ns
CLKOUT		21.3ns/54.0ns	23.5ns/54.8ns	24.1ns/57.0ns	26.5ns/57.0ns	25.0ns/58.7ns	22.1ns/59.2ns	24.6ns/60.1ns	24.4ns/58.7ns										13.2ns						
count5_clk		29.4ns	29.5ns	29.1ns	29.0ns	29.0ns	29.0ns	29.1ns	28.9ns	29.4ns	29.1ns	29.1ns	29.2ns	29.5ns	28.9ns	27.0ns	27.0ns								
count6_clk		26.5ns	26.6ns	26.3ns	28.9ns	26.6ns	26.6ns	28.7ns	26.2ns	26.7ns	27.9ns	26.4ns	26.5ns	26.8ns	29.0ns	28.9ns	28.9ns								
count7_clk		26.2ns	26.3ns	24.7ns	26.6ns	24.6ns	24.6ns	26.9ns	27.3ns	24.9ns	24.5ns	26.1ns	24.5ns	24.6ns	24.3ns	24.3ns	24.2ns								
count8_clk		29.2ns	29.3ns	27.7ns	24.5ns	26.2ns	26.2ns	24.6ns	25.1ns	26.1ns	27.7ns	25.0ns	26.8ns	27.5ns	25.7ns	25.7ns	23.1ns								
Data_in0																									
Data_in1																									
Data_in2																									
Data_in3																									
Data_in4																									
Data_in5																									
Data_in6																									
Data_in7																									
Data_in8																									
Data_in9																									
Data_in10																									
Data_in11																									
Data_in12																									
Data_in13																									
Data_in14																									
Data_in15																									
DB91_1_DCD																									
DB91_6_DSR																									
DB91_8_CTS																									
DB91_9_RI																									
DB92_1_DCD																							16.7ns		
DB92_4_DTR																							18.1ns		
DB92_6_DSR																							18.8ns		
DB92_8_CTS																					18.5ns				
DIPSW_8										24.7ns															
DIPSW_9											21.9ns														
DIPSW_10												21.8ns													
DIPSW_11													21.8ns												
DIPSW_12														21.8ns											
DIPSW_13															21.5ns										
DIPSW_14																21.5ns									
DIPSW_15																	23.7ns								
INST																								18.8ns	
P1RXD																									
P2_0																									
P2RXD																									
P5_NOZZEL										24.2ns															
P6_NOZZEL											21.3ns														
P7_NOZZEL												21.3ns													
P8_NOZZEL													24.3ns												
/RD		32.5ns/47.8ns	32.6ns/47.9ns	33.7ns/47.0ns	33.1ns/48.0ns	33.1ns/47.9ns	33.1ns/47.9ns	33.1ns/48.4ns	33.1ns/47.5ns	22.0ns/48.0ns	20.7ns/47.8ns	20.7ns/47.8ns	20.6ns/47.9ns	20.5ns/48.1ns	19.7ns/46.9ns	19.7ns/46.5ns	19.7ns/46.5ns								
RESET																									
/WR	27.0ns									34.6ns/42.1ns	34.6ns/39.3ns	34.6ns/39.1ns	34.6ns/39.1ns	34.6ns/41.4ns	34.3ns/41.7ns	34.3ns/41.7ns	33.9ns/41.8ns								19.5ns/33.4ns

Delay Matrix

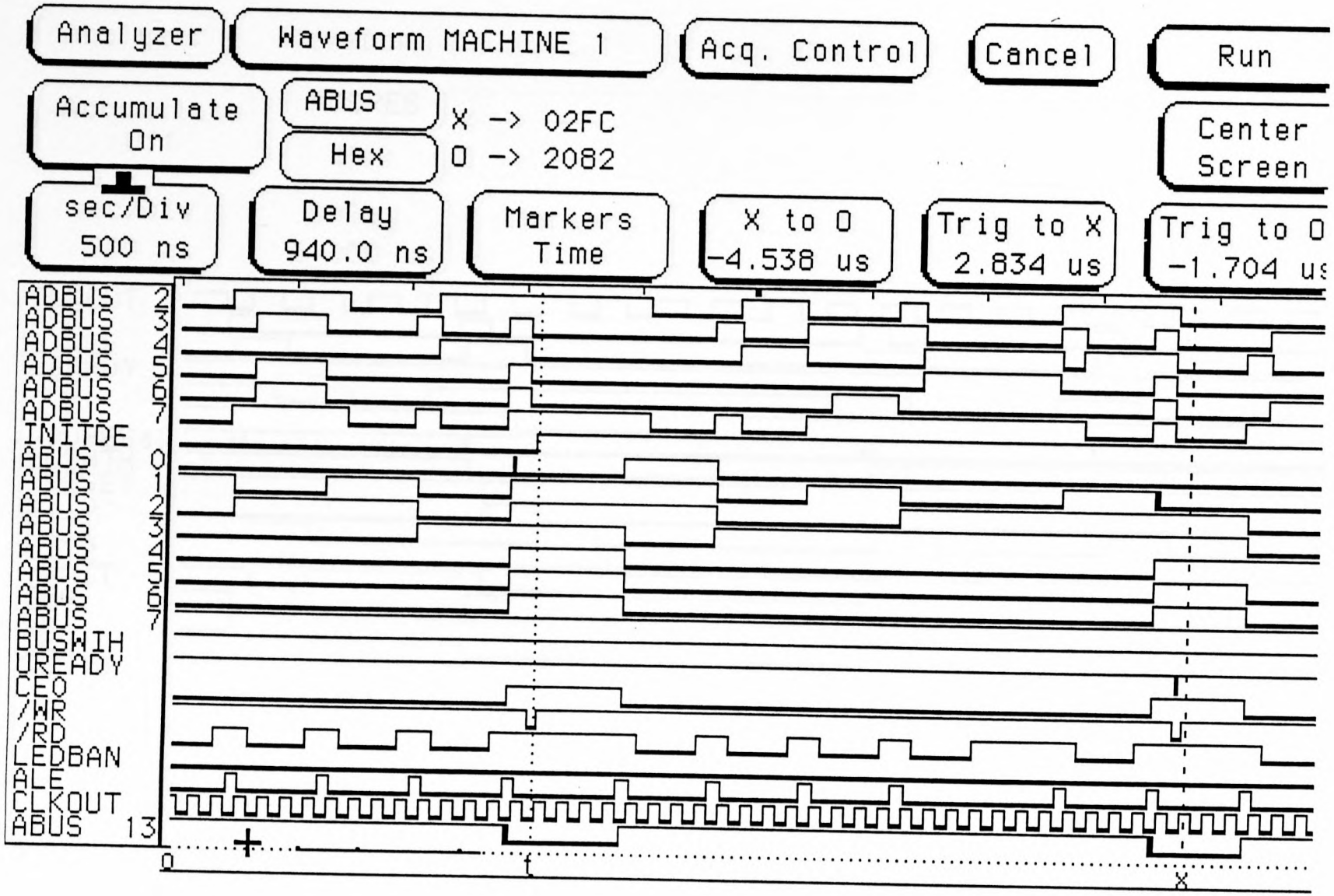
Destination

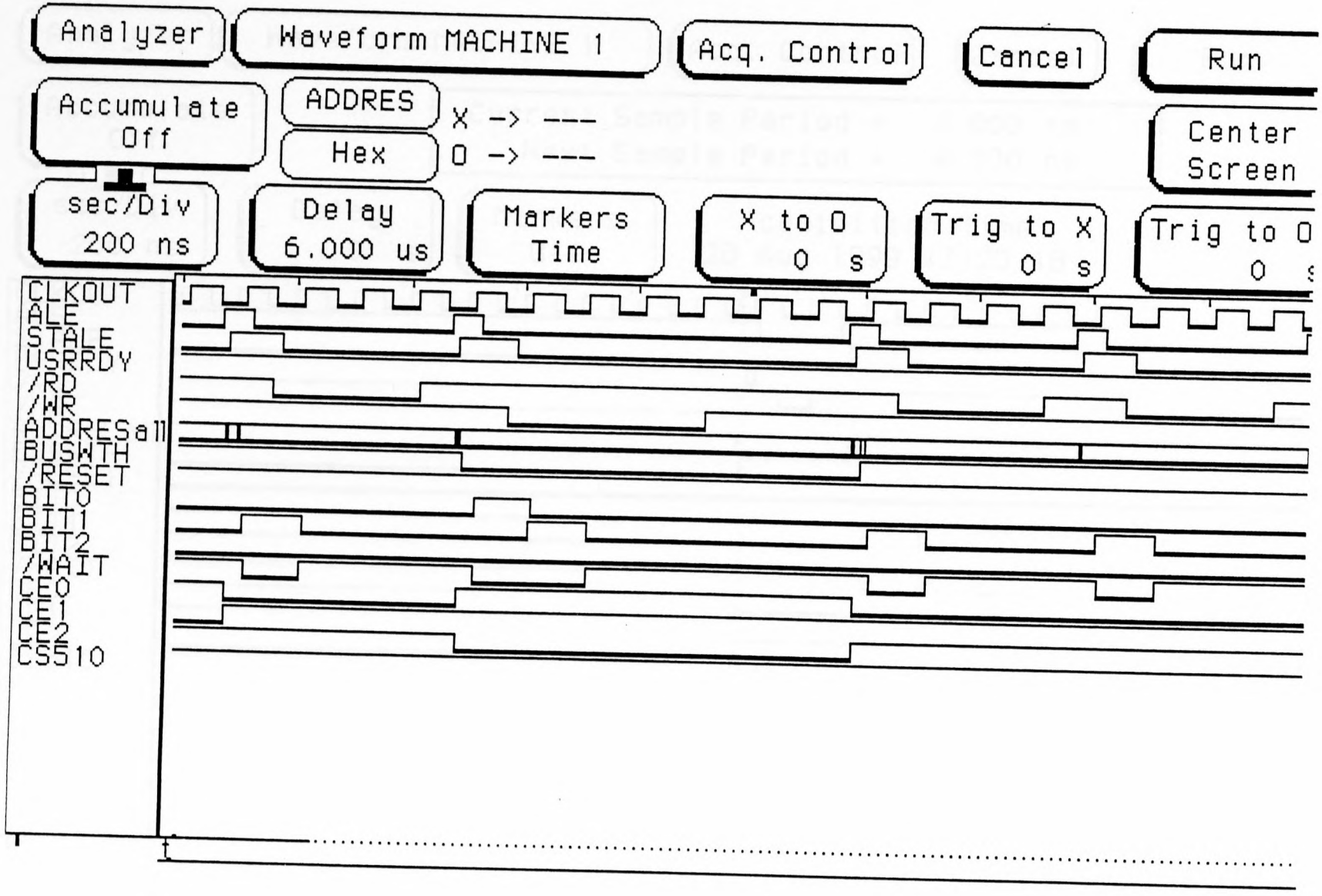
Source	PLED5A1	PLED5A2	PLED6A1	PLED6A2	PLED7A1	PLED7A2	PLED8A1	PLED8A2	P1TXD	P2_1	P2TXD	Rt.5	Rt.6	Rt.7	Rt.8	SPARE5	SPARE6	SPARE7	SPARE8	UTXD
ALE																				
A0																				
A1	26.0ns/30.6ns	26.0ns/30.6ns	26.9ns/30.7ns	26.4ns/30.7ns	26.2ns/30.7ns	26.3ns/30.7ns	26.3ns/30.7ns	22.5ns/30.7ns				26.2ns/30.3ns	26.2ns/30.3ns	26.2ns/30.3ns	26.2ns/30.3ns	26.2ns/30.3ns	26.3ns/30.2ns	26.3ns/30.2ns	24.8ns/30.2ns	
A2	27.4ns/32.0ns	27.4ns/32.0ns	27.4ns/32.1ns	27.8ns/32.1ns	27.6ns/32.1ns	27.7ns/32.1ns	27.7ns/32.1ns	23.9ns/32.1ns				28.5ns/32.6ns	28.5ns/32.6ns	28.5ns/32.6ns	28.5ns/32.6ns	28.5ns/32.6ns	28.6ns/32.5ns	28.6ns/32.5ns	27.1ns/32.5ns	
A3	27.4ns/32.0ns	27.4ns/32.0ns	27.4ns/32.1ns	27.8ns/32.1ns	27.6ns/32.1ns	27.7ns/32.1ns	27.7ns/32.1ns	23.9ns/32.1ns				28.3ns/32.4ns	28.3ns/32.4ns	28.3ns/32.4ns	28.3ns/32.4ns	28.3ns/32.4ns	28.4ns/32.3ns	28.4ns/32.3ns	26.9ns/32.3ns	
A4	21.6ns/26.2ns	21.6ns/26.2ns	21.6ns/26.3ns	22.0ns/26.3ns	21.8ns/26.3ns	21.9ns/26.3ns	21.9ns/26.3ns	18.1ns/26.3ns				21.6ns/25.7ns	21.6ns/25.7ns	21.6ns/25.7ns	21.6ns/25.7ns	21.6ns/25.7ns	21.7ns/25.6ns	21.7ns/25.6ns	20.2ns/25.6ns	
A5	21.7ns/26.3ns	21.7ns/26.3ns	21.7ns/26.4ns	22.1ns/26.4ns	21.9ns/26.4ns	22.0ns/26.4ns	22.0ns/26.4ns	18.2ns/26.4ns				21.1ns/25.2ns	21.1ns/25.2ns	21.1ns/25.2ns	21.1ns/25.2ns	21.1ns/25.2ns	21.2ns/25.1ns	21.2ns/25.1ns	19.7ns/25.1ns	
A6	32.4ns/37.0ns	32.4ns/37.0ns	32.4ns/37.1ns	32.8ns/37.1ns	32.6ns/37.1ns	32.7ns/37.1ns	32.7ns/37.1ns	28.9ns/37.1ns				33.0ns/37.1ns	33.0ns/37.1ns	33.0ns/37.1ns	33.0ns/37.1ns	33.0ns/37.1ns	33.1ns/37.0ns	33.1ns/37.0ns	31.6ns/37.0ns	
A7	32.4ns/37.0ns	32.4ns/37.0ns	32.4ns/37.1ns	32.8ns/37.1ns	32.6ns/37.1ns	32.7ns/37.1ns	32.7ns/37.1ns	28.9ns/37.1ns				33.0ns/37.1ns	33.0ns/37.1ns	33.0ns/37.1ns	33.0ns/37.1ns	33.0ns/37.1ns	33.1ns/37.0ns	33.1ns/37.0ns	31.6ns/37.0ns	
A8	37.8ns/42.4ns	37.8ns/42.4ns	37.8ns/42.5ns	38.2ns/42.5ns	38.0ns/42.5ns	38.1ns/42.5ns	38.1ns/42.5ns	34.3ns/42.5ns				38.4ns/42.5ns	38.4ns/42.5ns	38.4ns/42.5ns	38.4ns/42.5ns	38.4ns/42.5ns	38.5ns/42.4ns	38.5ns/42.4ns	37.0ns/42.4ns	
A9	36.2ns/40.8ns	36.2ns/40.8ns	36.2ns/40.9ns	36.6ns/40.9ns	36.4ns/40.9ns	36.5ns/40.9ns	36.5ns/40.9ns	32.7ns/40.9ns				36.8ns/40.9ns	36.8ns/40.9ns	36.8ns/40.9ns	36.8ns/40.9ns	36.8ns/40.9ns	36.9ns/40.8ns	36.9ns/40.8ns	35.4ns/40.8ns	
A10	37.8ns/42.4ns	37.8ns/42.4ns	37.8ns/42.5ns	38.2ns/42.5ns	38.0ns/42.5ns	38.1ns/42.5ns	38.1ns/42.5ns	34.3ns/42.5ns				38.4ns/42.5ns	38.4ns/42.5ns	38.4ns/42.5ns	38.4ns/42.5ns	38.4ns/42.5ns	38.5ns/42.4ns	38.5ns/42.4ns	37.0ns/42.4ns	
A11	37.8ns/42.4ns	37.8ns/42.4ns	37.8ns/42.5ns	38.2ns/42.5ns	38.0ns/42.5ns	38.1ns/42.5ns	38.1ns/42.5ns	34.3ns/42.5ns				38.4ns/42.5ns	38.4ns/42.5ns	38.4ns/42.5ns	38.4ns/42.5ns	38.4ns/42.5ns	38.5ns/42.4ns	38.5ns/42.4ns	37.0ns/42.4ns	
A12	40.2ns/44.8ns	40.2ns/44.8ns	40.2ns/44.9ns	40.6ns/44.9ns	40.4ns/44.9ns	40.5ns/44.9ns	40.5ns/44.9ns	36.7ns/44.9ns				40.8ns/44.9ns	40.8ns/44.9ns	40.8ns/44.9ns	40.8ns/44.9ns	40.8ns/44.9ns	40.9ns/44.8ns	40.9ns/44.8ns	39.4ns/44.8ns	
A13	39.7ns/44.3ns	39.7ns/44.3ns	39.7ns/44.4ns	40.1ns/44.4ns	39.9ns/44.4ns	40.0ns/44.4ns	40.0ns/44.4ns	36.2ns/44.4ns				40.3ns/44.4ns	40.3ns/44.4ns	40.3ns/44.4ns	40.3ns/44.4ns	40.3ns/44.4ns	40.4ns/44.3ns	40.4ns/44.3ns	38.9ns/44.3ns	
A14	40.2ns/44.8ns	40.2ns/44.8ns	40.2ns/44.9ns	40.6ns/44.9ns	40.4ns/44.9ns	40.5ns/44.9ns	40.5ns/44.9ns	36.7ns/44.9ns				40.8ns/44.9ns	40.8ns/44.9ns	40.8ns/44.9ns	40.8ns/44.9ns	40.8ns/44.9ns	40.9ns/44.8ns	40.9ns/44.8ns	39.4ns/44.8ns	
A15	43.8ns/48.4ns	43.8ns/48.4ns	43.8ns/48.5ns	44.2ns/48.5ns	44.0ns/48.5ns	44.1ns/48.5ns	44.1ns/48.5ns	40.3ns/48.5ns				44.4ns/48.5ns	44.4ns/48.5ns	44.4ns/48.5ns	44.4ns/48.5ns	44.4ns/48.5ns	44.5ns/48.4ns	44.5ns/48.4ns	43.0ns/48.4ns	
CLKOUT									19.0ns/19.3ns											
count5_clk																				
count6_clk																				
count7_clk																				
count8_clk																				
Data.in0																				
Data.in1																				
Data.in2																				
Data.in3																				
Data.in4																				
Data.in5																				
Data.in6																				
Data.in7																				
Data.in8	16.9ns											16.0ns								
Data.in9			16.9ns										16.0ns							
Data.in10					16.8ns									15.6ns						
Data.in11							16.7ns								15.6ns					
Data.in12		16.8ns														15.9ns				
Data.in13				16.7ns													15.9ns			
Data.in14						16.6ns												15.9ns		
Data.in15								16.1ns											16.0ns	
DB91_1_DCD																				
DB91_6_DSR																				
DB91_8_CTS																				
DB91_9_RI																				
DB92_1_DCD																				
DB92_4_DTR																				
DB92_6_DSR																				
DB92_8_CTS																				
DIPSW_8																				
DIPSW_9																				
DIPSW_10																				
DIPSW_11																				
DIPSW_12																				
DIPSW_13																				
DIPSW_14																				
DIPSW_15																				
INST																				
P1RXD																				
P2_0											12.4ns									
P2RXD										15.5ns										
P5_NOZZEL																				
P6_NOZZEL																				
P7_NOZZEL																				
P8_NOZZEL																				
/RD																				
RESET																				
/WR	20.5ns/35.3ns	20.5ns/35.3ns	20.6ns/35.4ns	20.6ns/35.4ns	20.6ns/35.4ns	20.6ns/35.4ns	20.6ns/35.4ns	20.6ns/35.4ns				20.0ns/35.4ns	20.0ns/35.4ns	20.0ns/35.4ns	20.0ns/35.4ns	20.0ns/35.4ns	19.9ns/35.3ns	19.9ns/35.3ns	19.9ns/35.3ns	

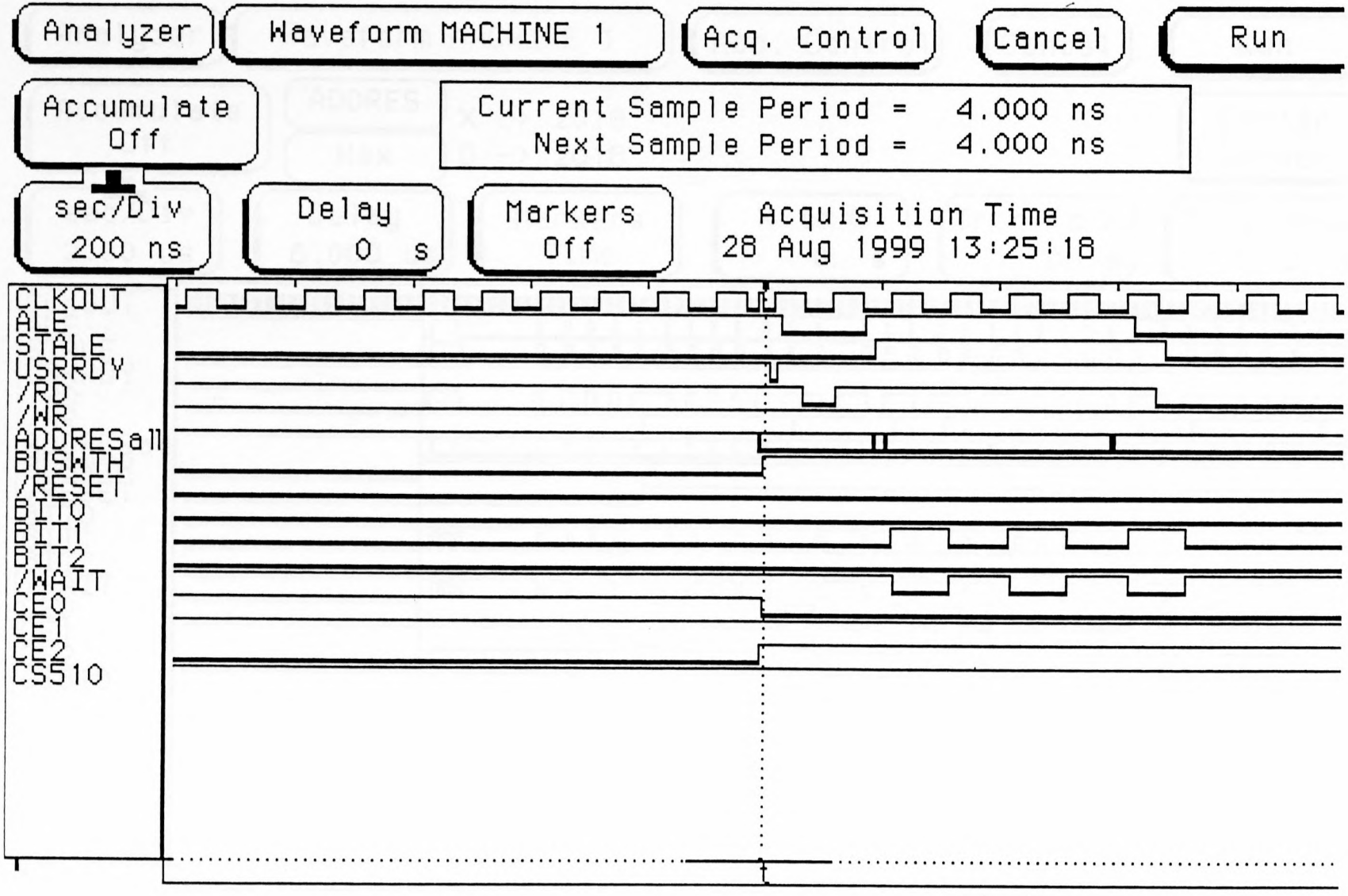
Appendix T

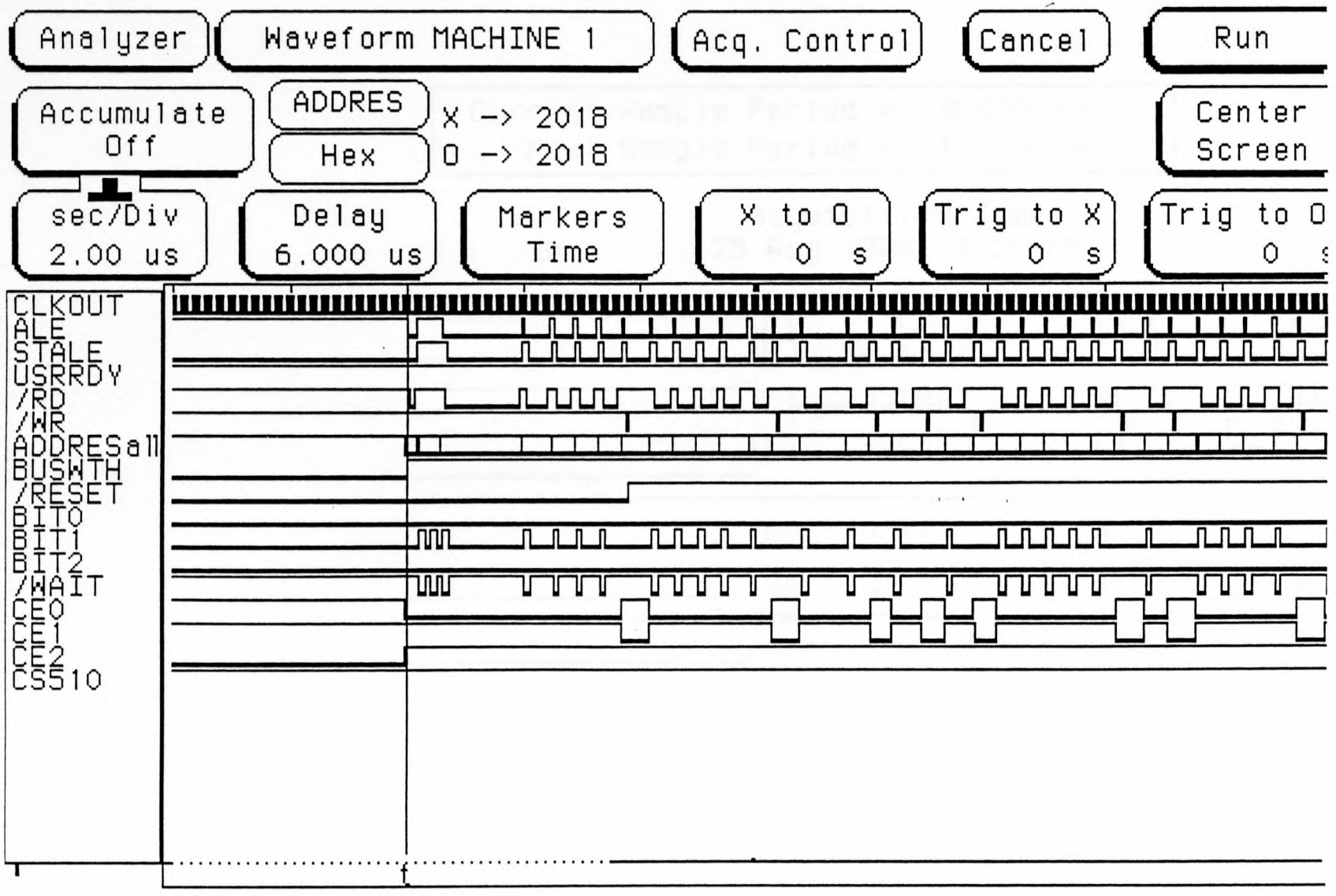
Hardware Timing Analysis using the Logic Analyzer

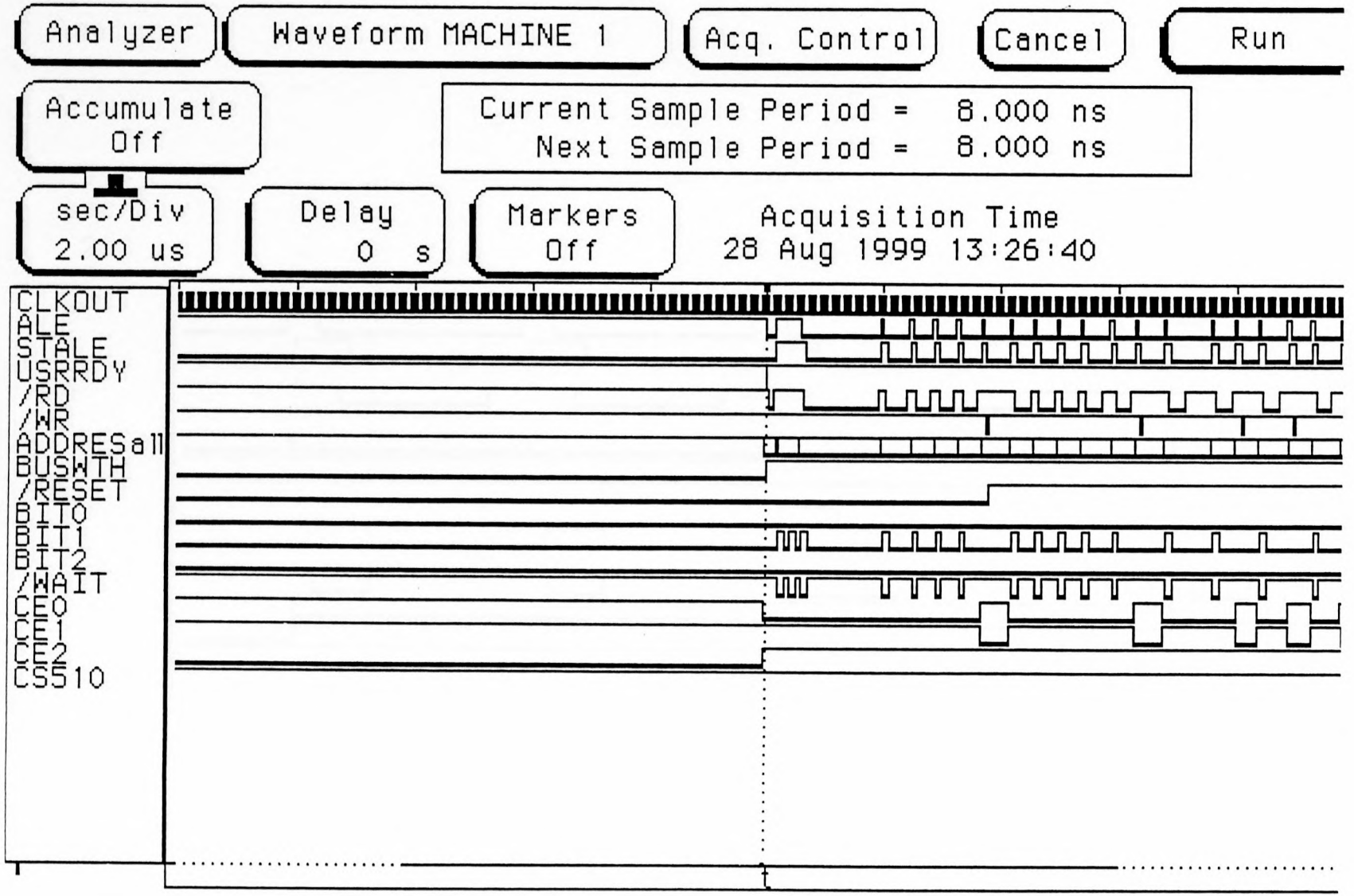
untitled - 1


Analyzer
Waveform MACHINE 1
Acq. Control
Cancel
Run
Accumulate On
ABUS
Hex
X -> 02FC
O -> 2082
Center Screen
sec/Div 500 ns
Delay 940.0 ns
Markers Time
X to O -4.538 us
Trig to X 2.834 us
Trig to O -1.704 us
ADBUS 2
ADBUS 3
ADBUS 4
ADBUS 5
ADBUS 6
ADBUS 7
INITDE
ABUS 0
ABUS 1
ABUS 2
ABUS 3
ABUS 4
ABUS 5
ABUS 6
ABUS 7
BUSWIH
UREADY
CEO
/WR
/RD
LEDBAN
ALE
CLKOUT
ABUS 13
O
X

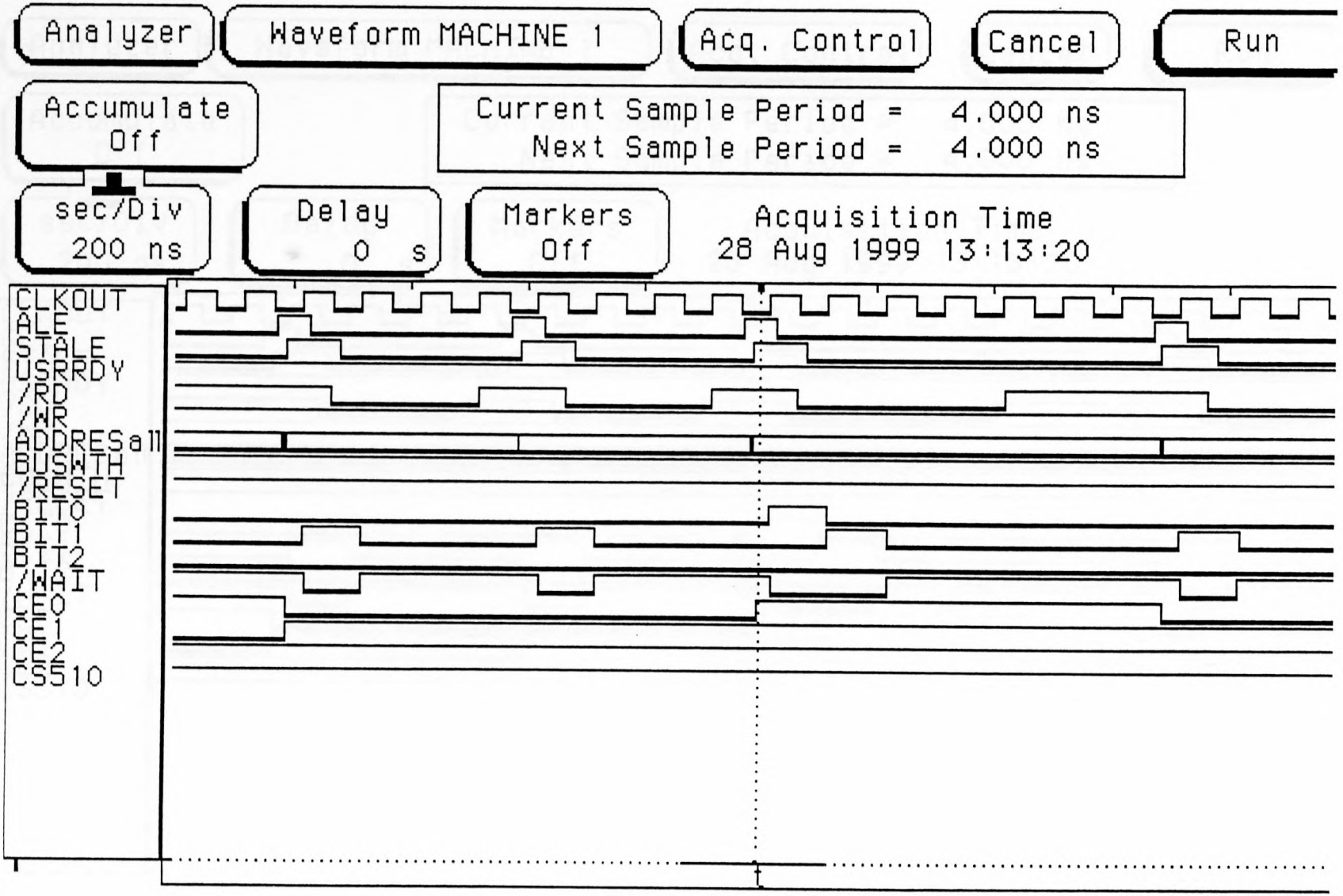

Analyzer
Waveform MACHINE 1
Acq. Control
Cancel
Run
Accumulate Off
ADDRES
Hex
X -> ---
O -> ---
Center Screen
sec/Div 200 ns
Delay 6.000 us
Markers Time
X to O 0 s
Trig to X 0 s
Trig to O 0
CLKOUT
ALE
STALE
USRRDY
/RD
/WR
ADDRESall
BUSWTH
/RESET
BIT0
BIT1
BIT2
/WAIT
CE0
CE1
CE2
CS510

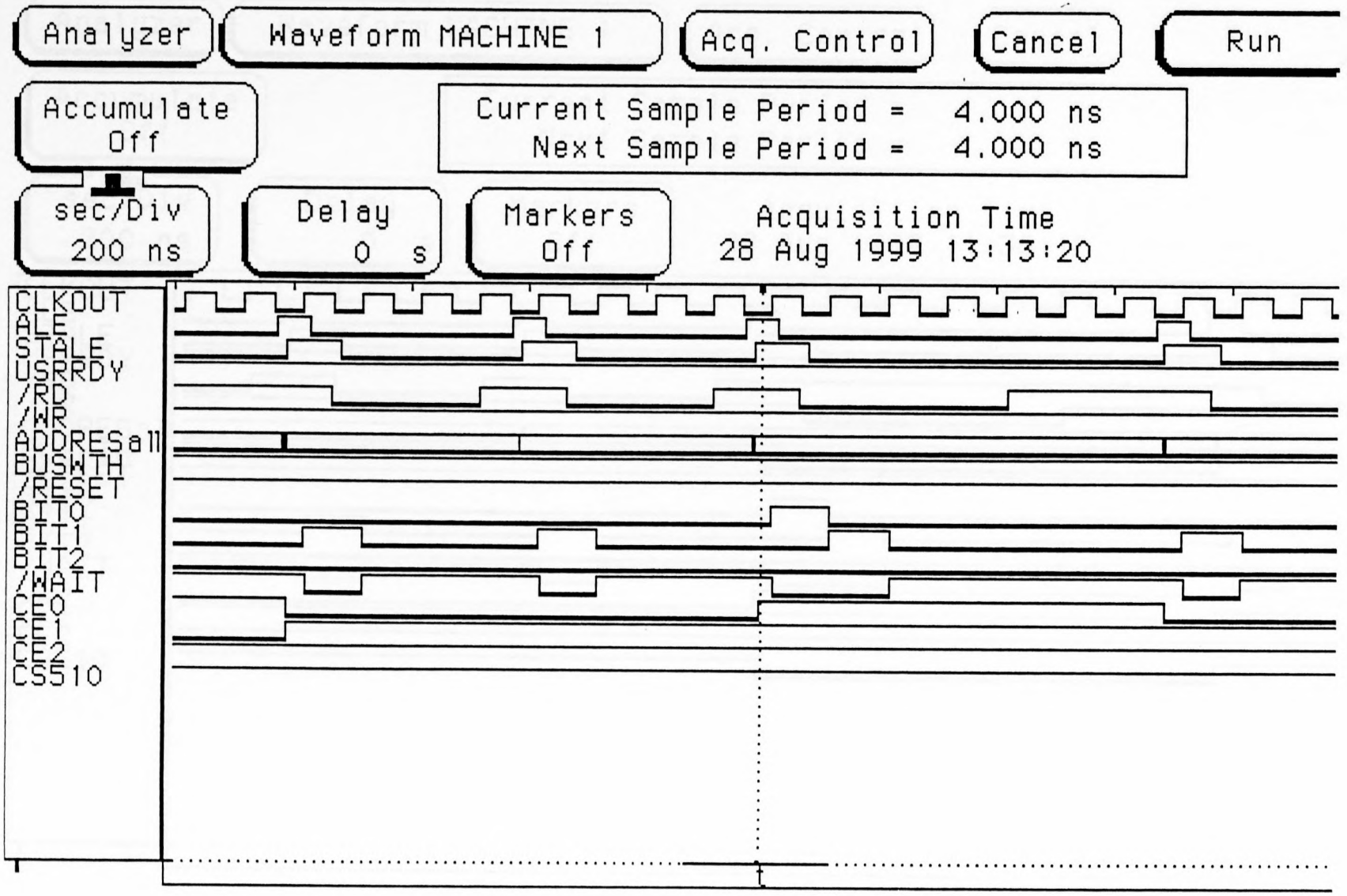

Analyzer
Waveform MACHINE 1
Acq. Control
Cancel
Run
Accumulate
Off
Current Sample Period = 4.000 ns
Next Sample Period = 4.000 ns
sec/Div
200 ns
Delay
0 s
Markers
Off
Acquisition Time
28 Aug 1999 13:25:18
CLKOUT
ALE
STALE
USRRDY
/RD
/WR
ADDRESall
BUSWTH
/RESET
BIT0
BIT1
BIT2
/WAIT
CE0
CE1
CE2
CS510


Analyzer
Waveform MACHINE 1
Acq. Control
Cancel
Run
Accumulate Off
ADDRES
Hex
X -> 2018
0 -> 2018
Center Screen
sec/Div 2.00 us
Delay 6.000 us
Markers Time
X to O 0 s
Trig to X 0 s
Trig to O 0
CLKOUT
ALE
STALE
USRRDY
/RD
/WR
ADDRESall
BUSWTH
/RESET
BIT0
BIT1
BIT2
/WAIT
CE0
CE1
CE2
CS510

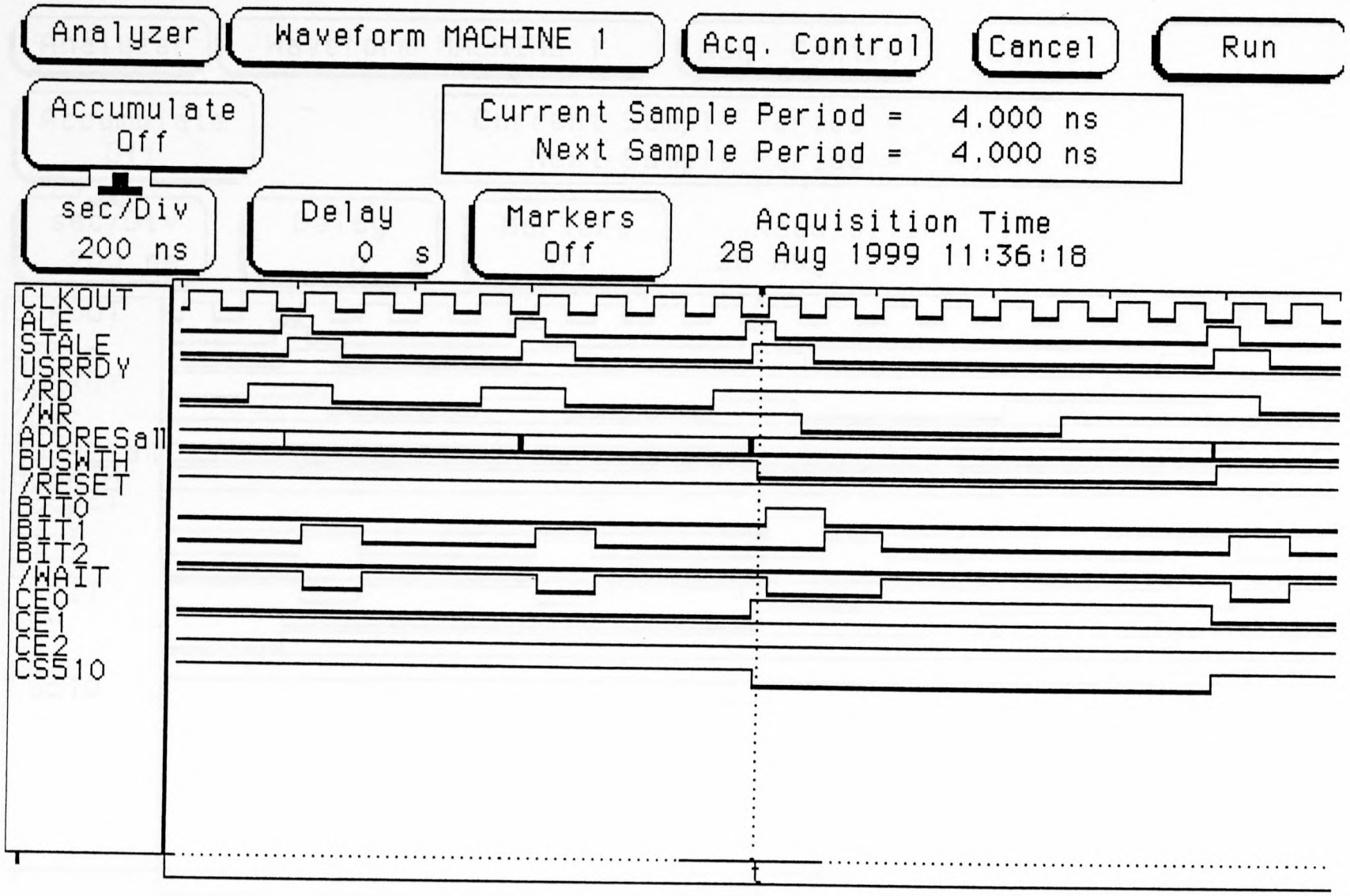

Analyzer
Waveform MACHINE 1
Acq. Control
Cancel
Run
Accumulate
Off
Current Sample Period = 8.000 ns
Next Sample Period = 8.000 ns
sec/Div
2.00 us
Delay
0 s
Markers
Off
Acquisition Time
28 Aug 1999 13:26:40
CLKOUT
ALE
STALE
USRRDY
/RD
/WR
ADDRESall
BUSWTH
/RESET
BIT0
BIT1
BIT2
/WAIT
CE0
CE1
CE2
CS510

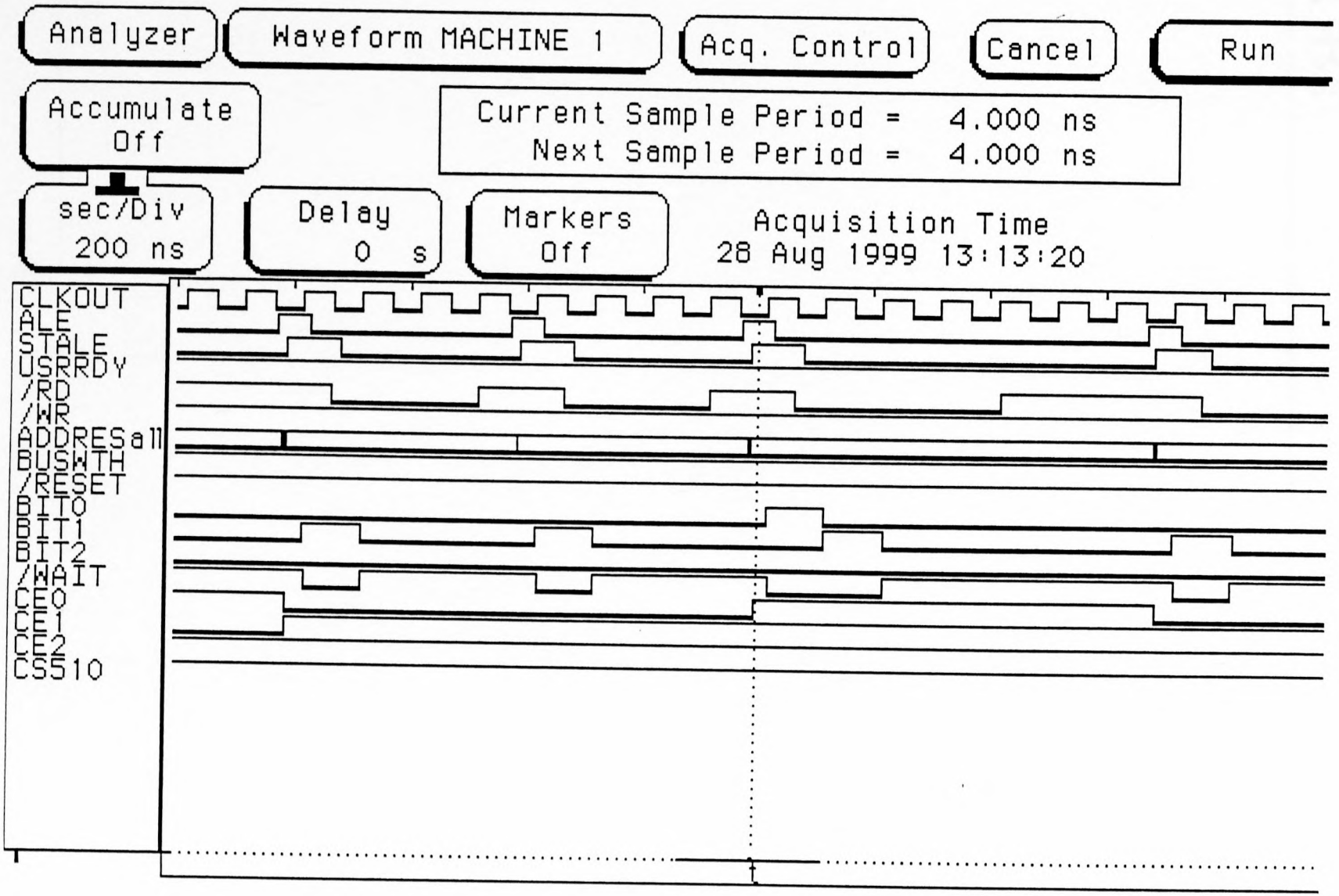
ce1_sel.tif


Analyzer
Waveform MACHINE 1
Acq. Control
Cancel
Run
Accumulate
Off
Current Sample Period = 4.000 ns
Next Sample Period = 4.000 ns
sec/Div
200 ns
Delay
0 s
Markers
Off
Acquisition Time
28 Aug 1999 13:13:20
CLKOUT
ALE
STALE
USRRDY
/RD
/WR
ADDRESall
BUSWTH
/RESET
BIT0
BIT1
BIT2
/WAIT
CE0
CE1
CE2
CS510


Analyzer
Waveform MACHINE 1
Acq. Control
Cancel
Run
Accumulate
Off
Current Sample Period = 4.000 ns
Next Sample Period = 4.000 ns
sec/Div
200 ns
Delay
0 s
Markers
Off
Acquisition Time
28 Aug 1999 13:13:20
CLKOUT
ALE
STALE
USRRDY
/RD
/WR
ADDRESall
BUSWTH
/RESET
BIT0
BIT1
BIT2
/WAIT
CE0
CE1
CE2
CS510

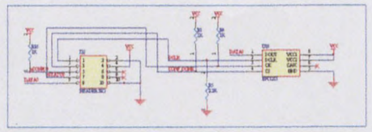
ph1_2ws.tif

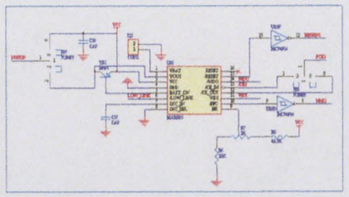

Analyzer
Waveform MACHINE 1
Acq. Control
Cancel
Run
Accumulate
Off
Current Sample Period = 4.000 ns
Next Sample Period = 4.000 ns
sec/Div
200 ns
Delay
0 s
Markers
Off
Acquisition Time
28 Aug 1999 11:36:18
CLKOUT
ALE
STALE
USRRDY
/RD
/WR
ADDRESall
BUSWTH
/RESET
BIT0
BIT1
BIT2
/WAIT
CE0
CE1
CE2
CS510

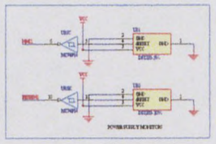

Analyzer
Waveform MACHINE 1
Acq. Control
Cancel
Run
Accumulate
Off
Current Sample Period = 4.000 ns
Next Sample Period = 4.000 ns
sec/Div
200 ns
Delay
0 s
Markers
Off
Acquisition Time
28 Aug 1999 13:13:20
CLKOUT
ALE
STALE
USRRDY
/RD
/WR
ADDRESall
BUSWTH
/RESET
BIT0
BIT1
BIT2
/WAIT
CE0
CE1
CE2
CS510

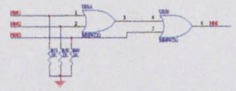
Appendix U

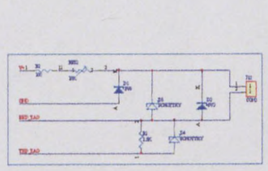
PHASE 2 Design Schematics and PCBs

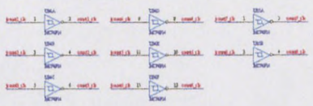


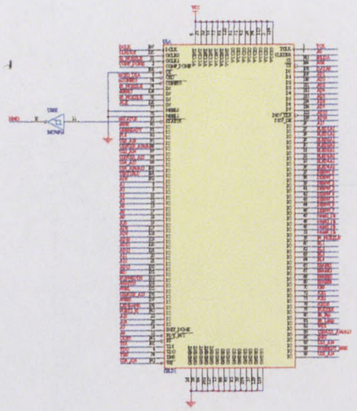


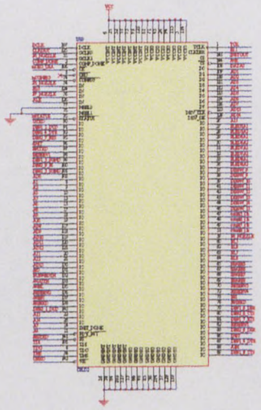


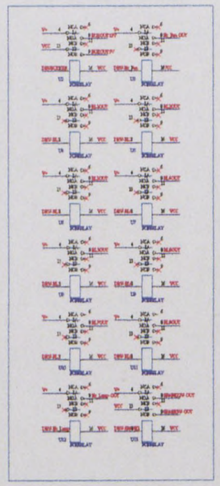


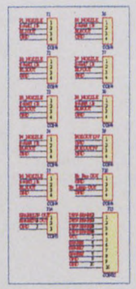


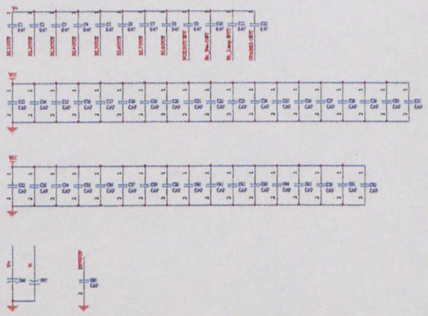


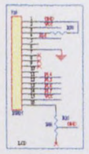


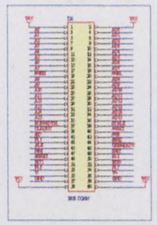


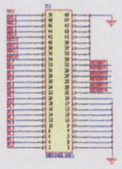


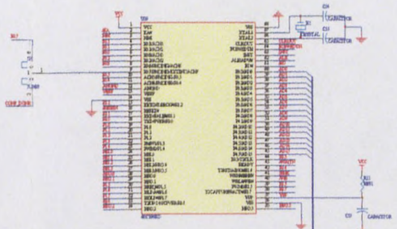


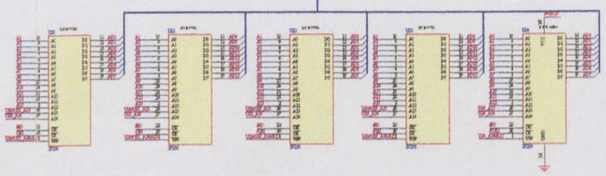


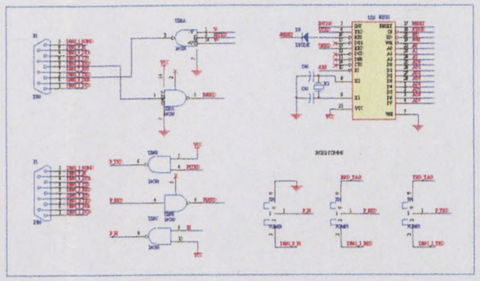


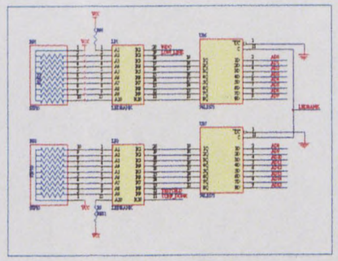


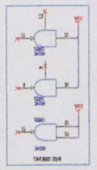


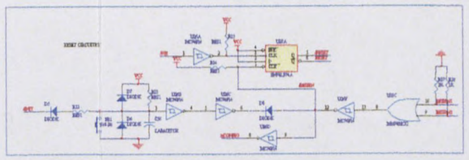


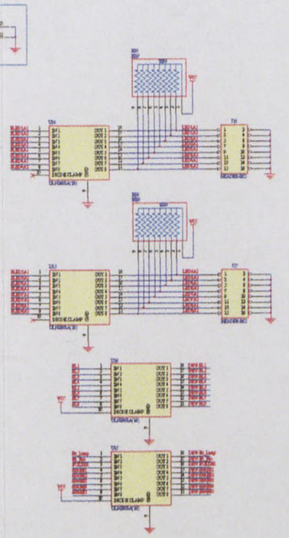


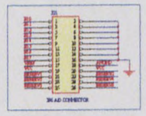


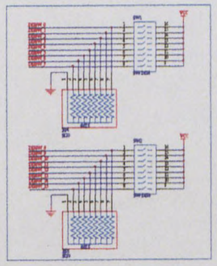


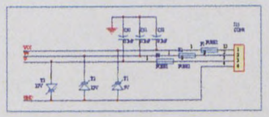


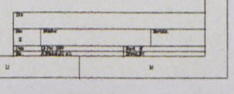


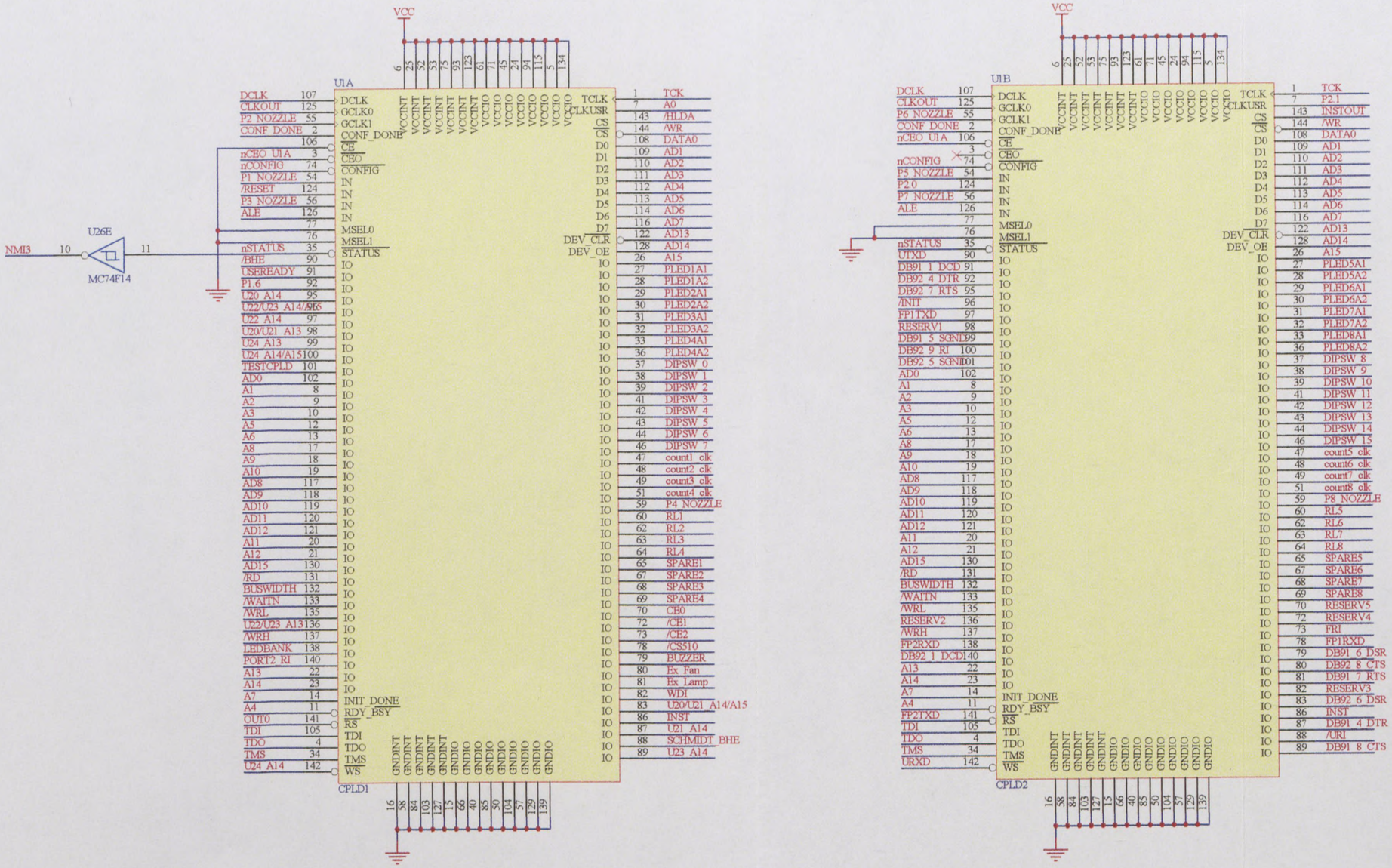




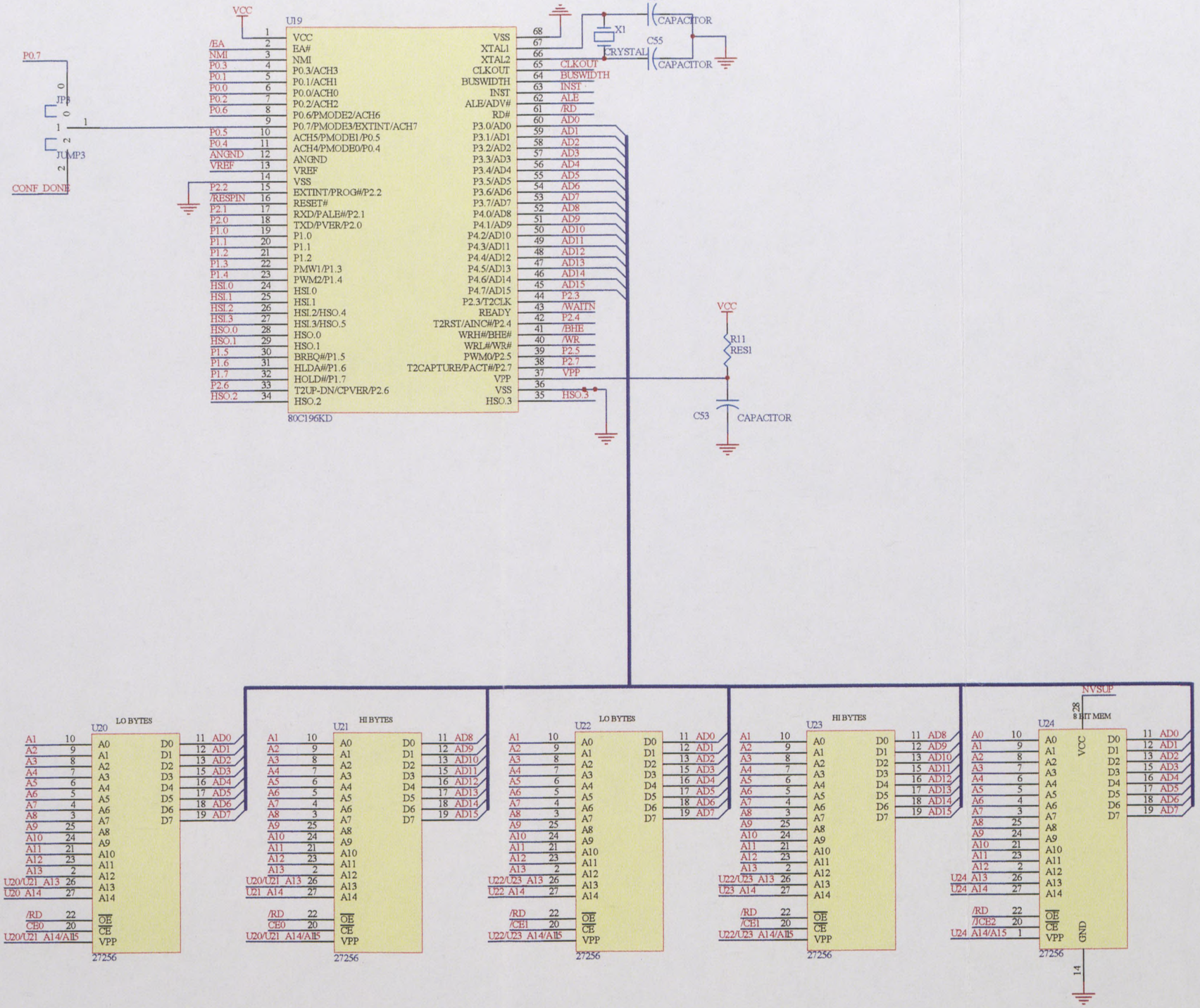




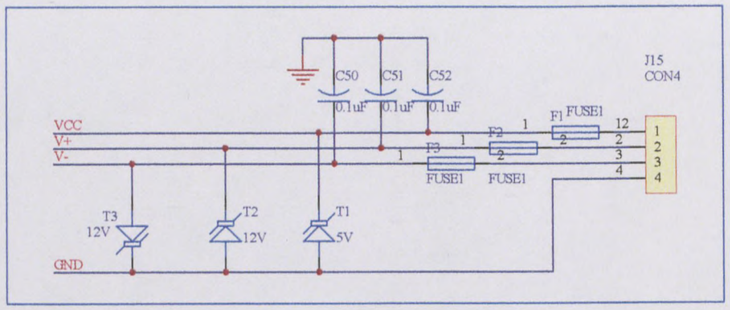


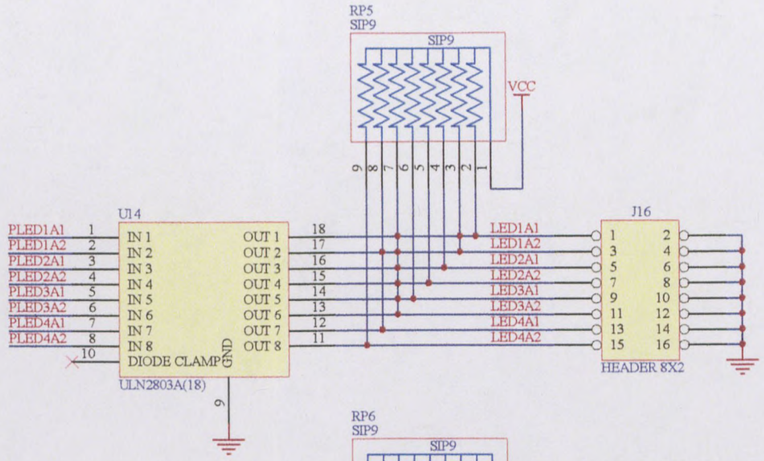


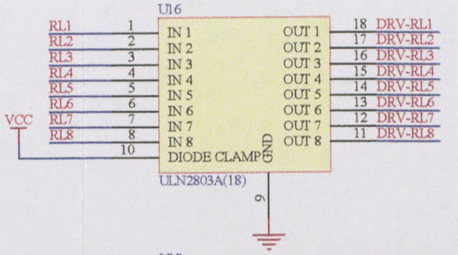
Title		
Size: A3	Number	Revision
Date: 12-Dec-1999	Sheet of	
File: D:\temp\cplds.sch	Drawn By:	

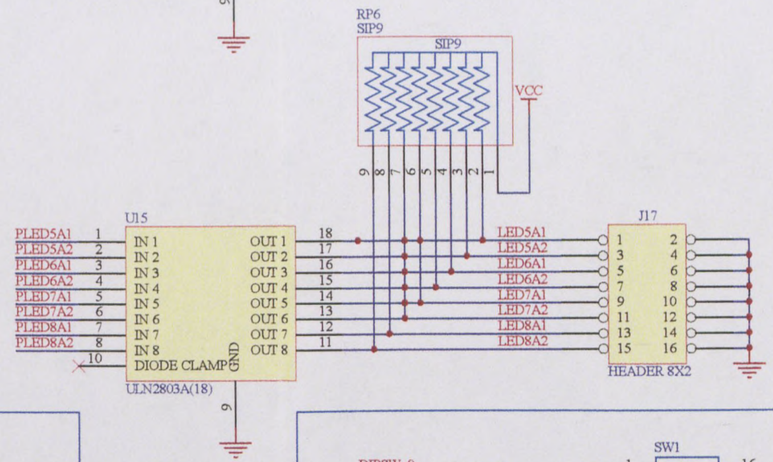


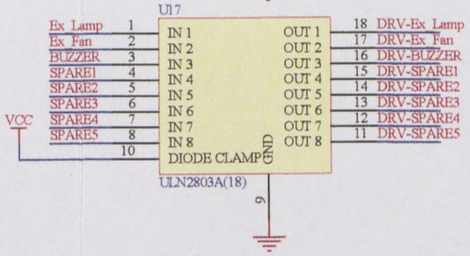
Title	MEMORY INTERFACE FOR THE 80C196KD MICROCONTROLLER	
Size: A3	Number: 1	Revision: 1.1
Date: 13-Dec-1999	Sheet 1of 1	
File: D:\temp\MICRO196.sch	Drawn By: STEVEN SWANEPOEL	

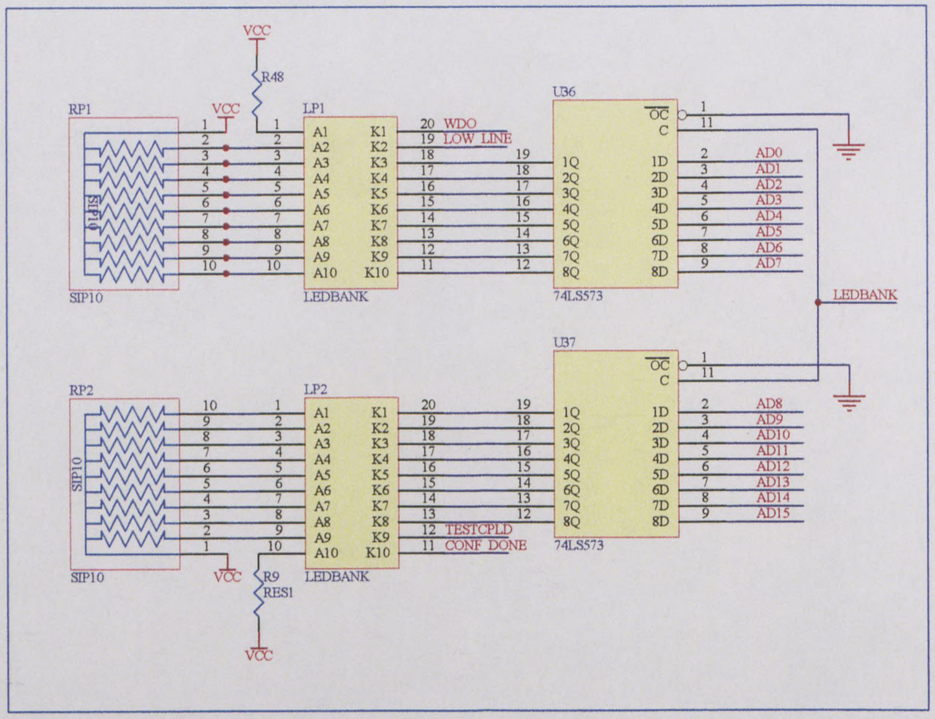


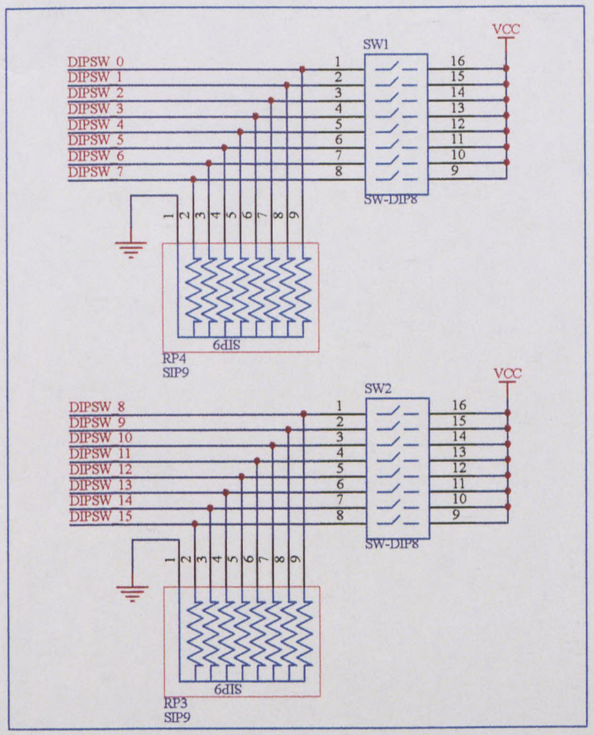




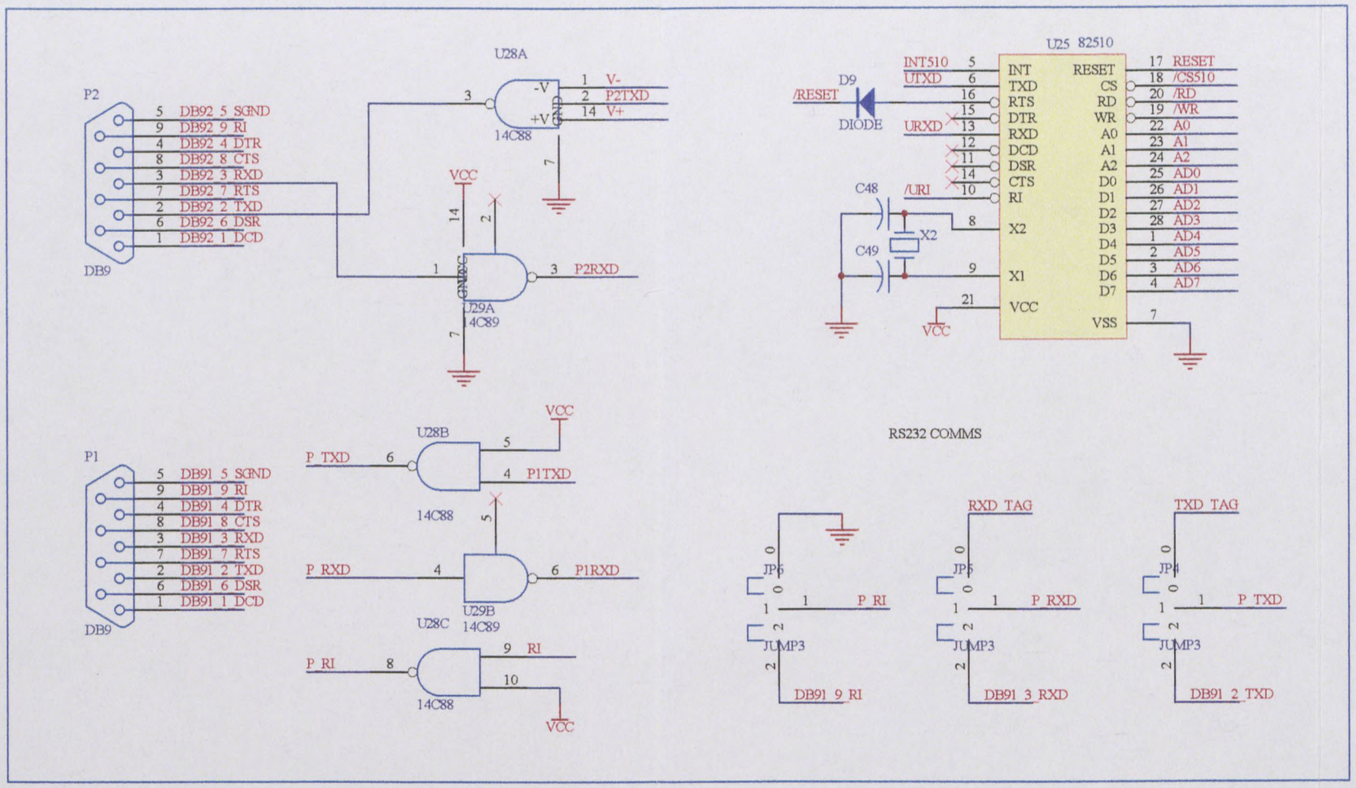


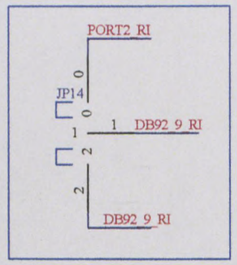


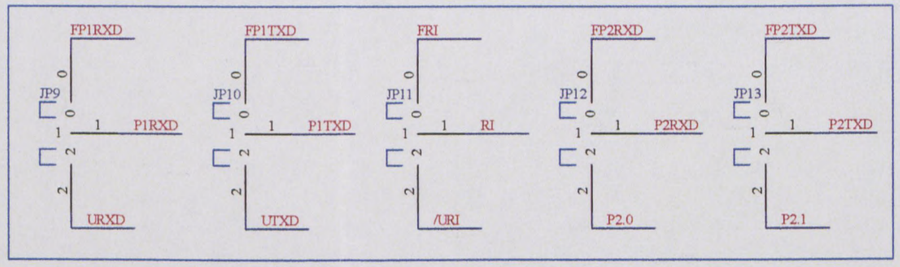




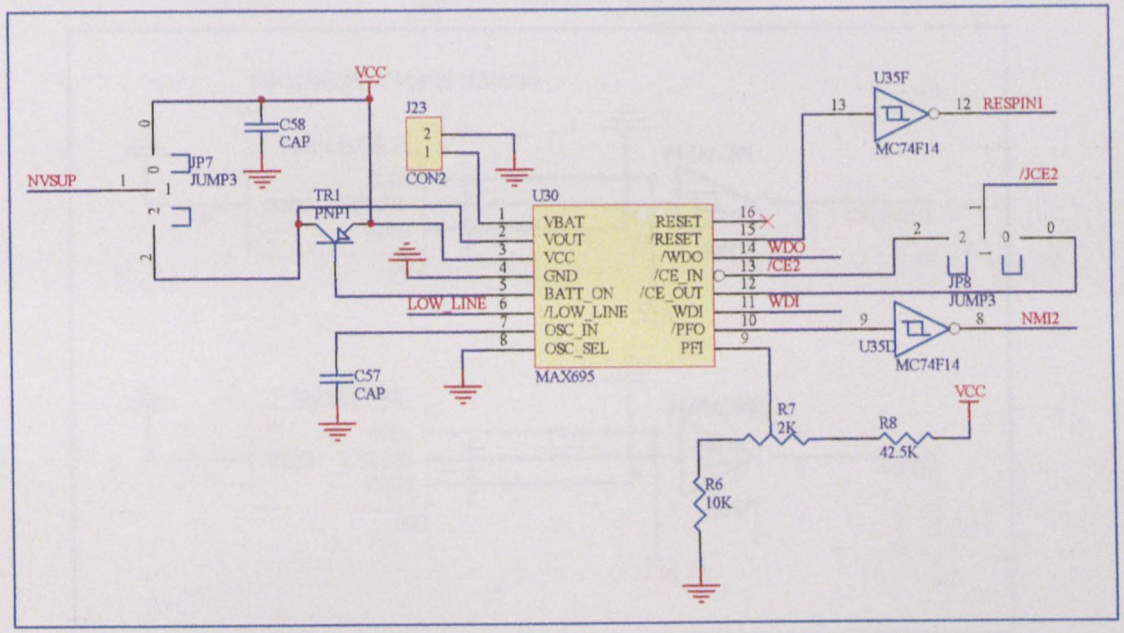
Title: PERIPHERAL CIRCUITRY		
Size: A3	Number: 1	Revision: 1.0
Date: 13-Dec-1999	Sheet 1of 1	
File: D:\temp\PERIPH1.sch	Drawn By: STEVEN SWANEPOEL	



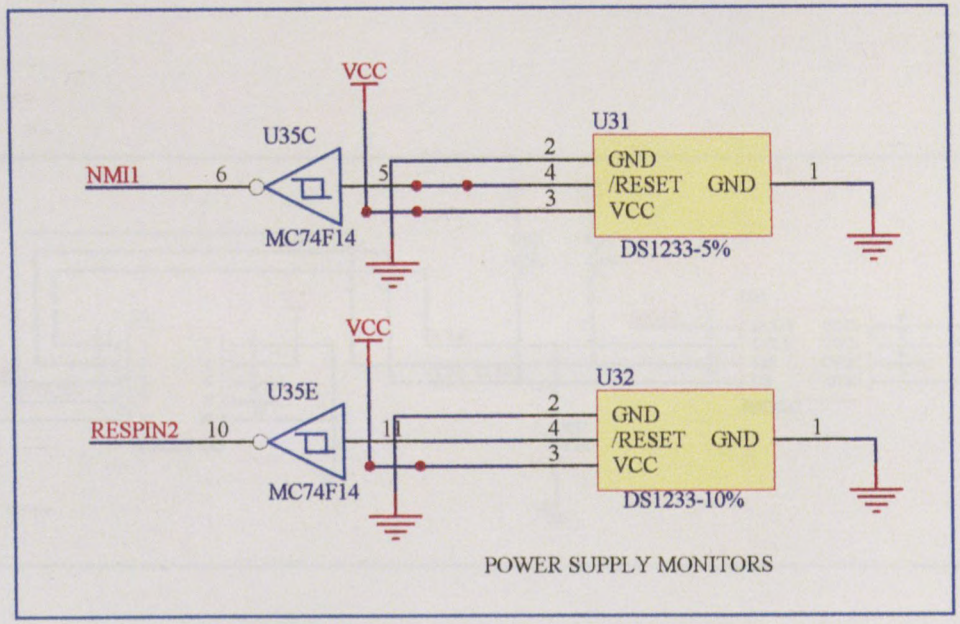




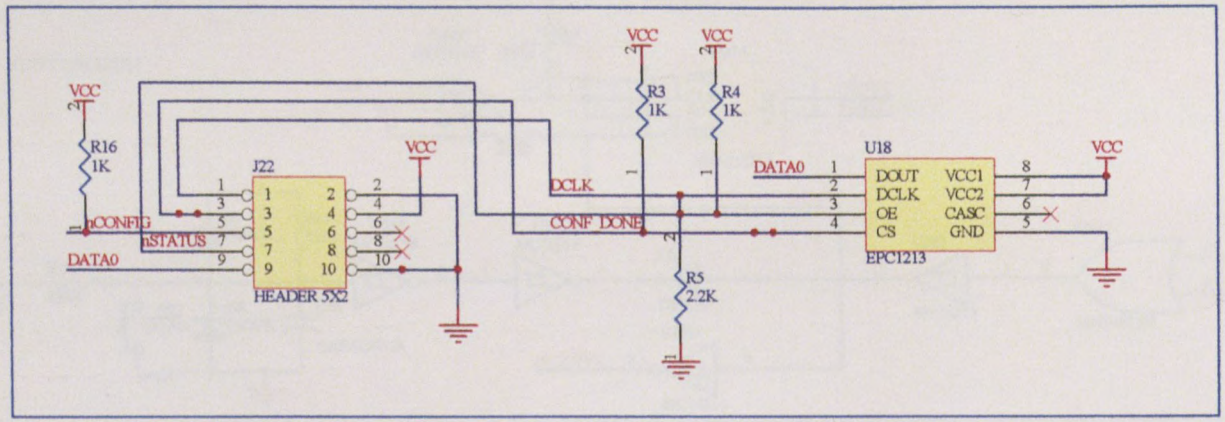
Title: EXTERNAL 82510 UART CONNECTIONS		
Size: A3	Number: 1	Revision: 1.0
Date: 13-Dec-1999		Sheet 1of 1
File: D:\temp\UART.sch		Drawn By: STEVEN SWANEPOEL



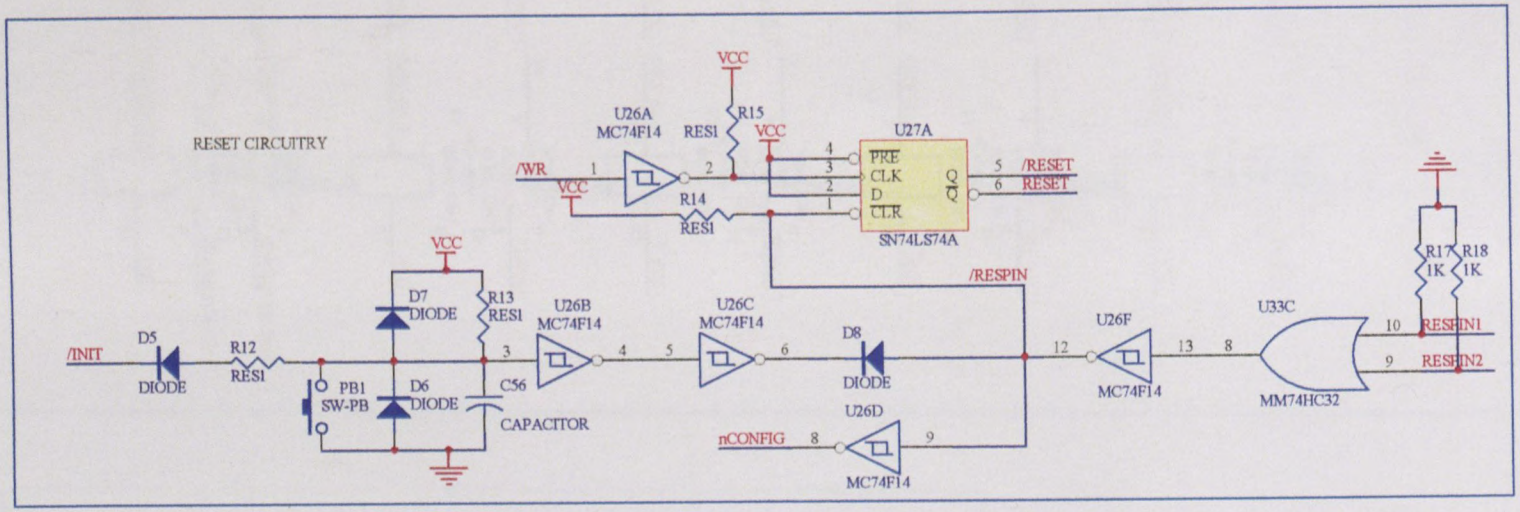
Title: POWER LOSS PROTECTION CIRCUITRY			
Size: A4	Number: 1		Revision: 1.1
Date: 12-Dec-1999		Sheet lof 1	
File: D:\temp\POWLOSS2.sch		Drawn By: STEVEN SWANEPOEL	



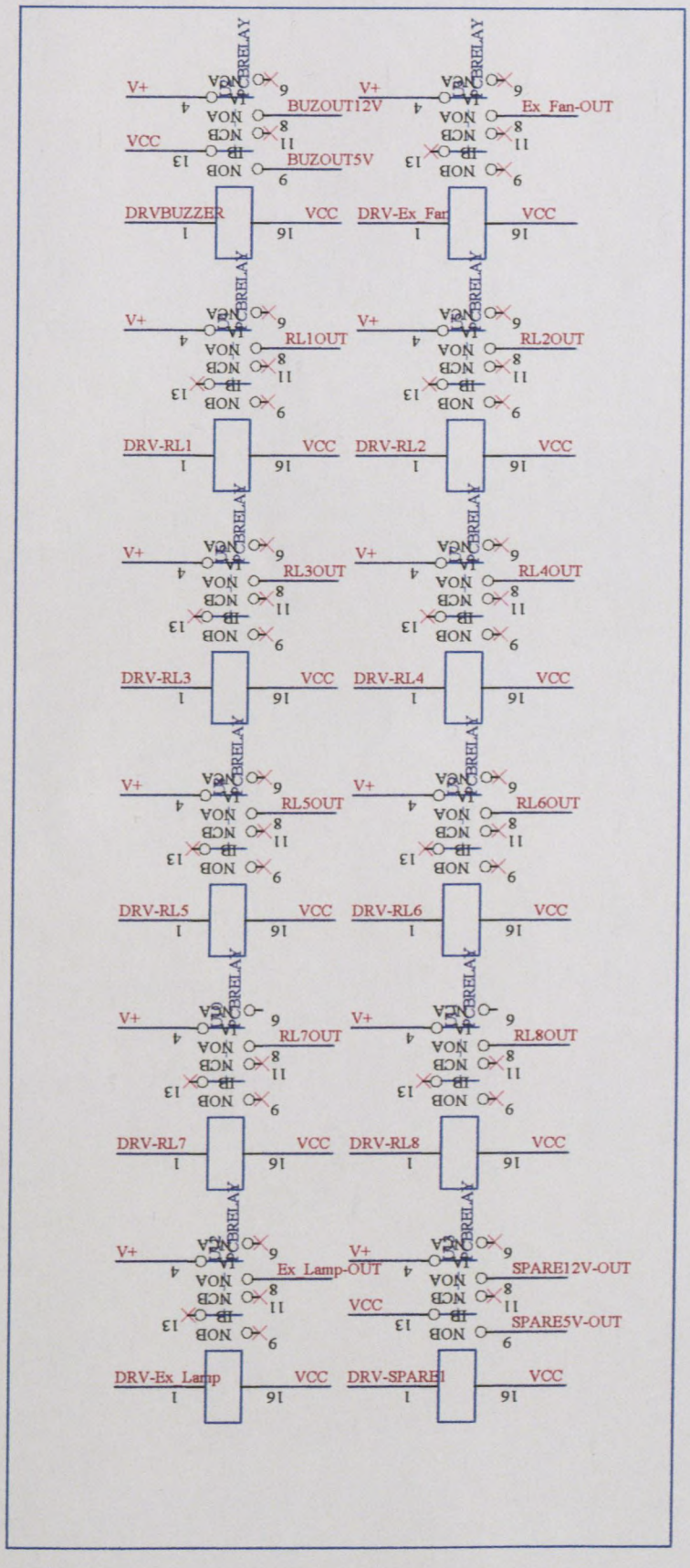
Title	POWER MONITORING CIRCUITRY		
Size A	Number 1		Revision 1.0
Date:	12-Dec-1999	Sheet 1of 1	
File:	D:\temp\powloss.sch	Drawn By:	STEVEN SWANEPOEL

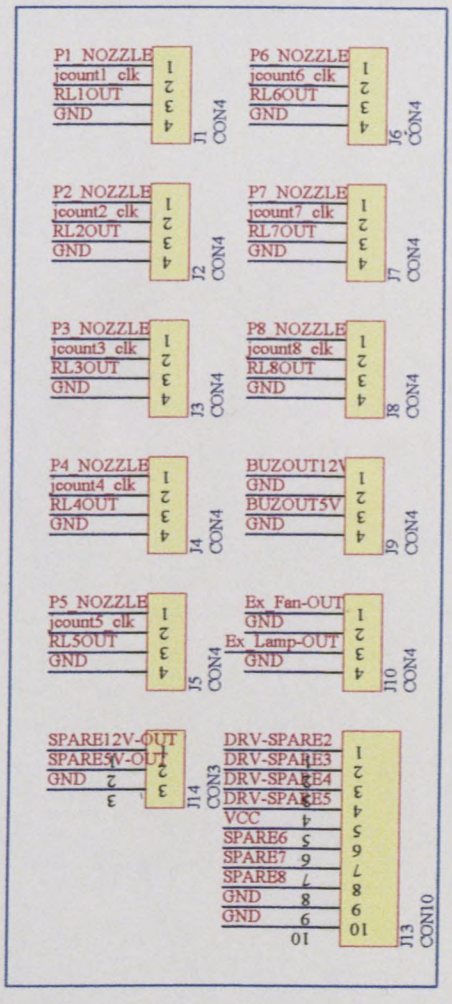


Title: CPLD CONFIGURATION CIRCUITRY		
Size: A4	Number: 1	Revision: 1.2
Date: 12-Dec-1999	Sheet 1of 1	
File: D:\temp\CONFIGCIR.sch	Drawn By: STEVEN SWANEPOEL	



Title: SYSTEM RESET CIRCUITRY			
Size: A4	Number: 1		Revision: 1.0
Date: 12-Dec-1999		Sheet 1of 1	
File: D:\temp\SYSRESET.sch		Drawn By: STEVEN SWANEPOEL	





Title	PERIPHERAL CIRCUITRY		
Size	Number	Revision	
A4	1	1.0	
Date:	13-Dec-1999	Sheet of	Text
File:	D:\temp\PERIPH2.sch	Drawn By:	STEVEN SWANEPOEL